VŠB TECHNICKÁ UNIVERZITA OSTRAVA
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Analýza velkých dat

Big Data Analysis

Jakub Szlaur

Bakalářská práce

Vedoucí práce: doc. Ing. Jan Žídek, CSc.

Ostrava, 2021

Abstrakt

Tato **bakalářská** práce je zaměřena na problematiku analýzy velkých dat v kontextu čtvrté průmyslové revoluce. Cílem práce je popsat celý proces zpracování velkých dat a následné otestování popsaných metod. Práce bude obsahovat **pojednání o tom**, jak se velká data sbírají a k čemu slouží. Dále bude tato práce pojednávat o jednotlivých nástrojích pro analýzu velkých dat**. Praktická část bakalářské práce je věnována** praktickému otestování těchto nástrojů na konkrétním datasetu.

Klíčová slova

Velká data, Průmysl 4.0, nástroje, dataset, analýza, strojové učení, modely

Abstract

This work is focused on big data analysis in the context of the fourth industrial revolution. The goal of the work is to describe the whole process of working with big data and testing the described methods. Work describes how big data are collected and how they can be used. This work also discusses the individual tools for analyzing big data and practical testing of those tools on a specific dataset.

Key words

Big data, Industry 4.0, tools, dataset, analysis, machine learning, models

Poděkování

Rád bych poděkoval vedoucímu bakalářské práce **doc. Ing. Jan Žídek, CSc.** za odbornou pomoc a konzultaci při vytváření této bakalářské práce. Dále bych také chtěl poděkovat své rodině a svým blízkým, kteří mě při vytváření této práce podporovali.

Obsah

Seznam použitých zkratek

Zkratka	Význam
AMQP	Advanced Message Queuing Protocol
OSI	Open Systems Interconnection
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
SCTP	Stream Control Transmission Protocol
XML	eXtensible Markup Language
JMS	Java Message Service
JSON	JavaScript Object Notation
HTTP	Hypertext Transfer Protocol
HDFS	Hadoop Distributed File System
WEKA	Waikato Environment for Knowledge Analysis
API	Application Programming Interface
JVM	Java Virtual Machine
CPU	Central Processing Unit
SQL	Structured Query Language
NoSQL	Not Only Structured Query Language
CSV	Comma Separated Values
TSV	Tab Separated Values
ACID	Atomicity, Consistency, Isolation, Durability
OLTP	Online Transactional Processing
AWS	Amazon Web Services
PCA	Principal Component Analysis

ICA	Independent Component Analysis
MCC	Matthews correlation coefficient
NaN	Not a Number
DAG	Directed acyclic graph

Seznam ilustrací a tabulek

Úvod

Velká data jsou novodobým fenoménem popisujícím analýzu velkého množství strukturovaných a nestrukturovaných dat. Využití velkých dat můžeme zaznamenat na více úrovních naší společnosti, ať už se bavíme o ekonomice, financích, bankovnictví, zdravotnictví, dopravě nebo o čtvrté průmyslové revoluci. I když se jedná o relativně novou disciplínu, jelikož dnes denně dochází ke generování obrovského množství nových dat.

První kapitola teoretické části práce se zabývá problematikou s definováním pojmu velká data. Narážíme zde totiž na skutečnost, že oficiální a obecně akceptovaná definice prozatím neexistuje. Definice se prokazují jako nedostatečné a neúplné. V dalších kapitolách teoretické části čtenář nalezne kapitoly pojednávající o nástrojích, které jsou klíčové pro různé fáze zpracovávání velkých dat. Dále se dočte o nejnovějších trendech v nástrojích a metodách pro sběr dat. Také o nástrojích pro analýzu velkých dat, společně s databázemi a úložišti pro velká data. Nakonec jsou zde popsány algoritmy a metody pro analýzu velkých dat pomocí strojového učení. Pro lepší přehlednost jednotlivých algoritmů je strojové učení rozděleno do podkapitol strojové učení s učitelem a bez učitele. Teoretická část tak čtenáři nabízí výčet nejrůznějších nástrojů s popisem jejich výhod a využití. Cílem tedy není detailně vysvětlit každý nástroj a jeho interní fungování, ale pouze čtenáře obeznámit se základními koncepty a jejich hlavními charakteristikami.

Praktická část práce analyzuje konkrétní dataset vybraný z oblasti čtvrté průmyslové revoluce. Analýza je provedena pomocí zvolených softwarových nástrojů. Prozkoumávaný dataset byl vybrán ze stránek Kaggle a jedná se o vygenerovaný dataset produkční linky firmy Bosch. Jelikož se jedná o jeden z nejkomplexnějších datasetů, který můžeme na platformě Kaggle najít, lze otestovat široké portfolium postupů a metod na vizualizaci, transformaci, redukci a také analýzu velkých dat. Povaha zvoleného datasetu podtrhuje důležitost předkládaného projektu. Pro účely této práce je zvolen programovací jazyk Python a knihovny jako jsou sklearn, matplotlib, networkx, pandas nebo dask. JupyterLab je prostředí, ve kterém jsou vypracovány zdrojové kódy, s využitím distribuce Anaconda.

Jelikož dataset obsahuje data numerická, kategorická a také časová, je praktická část rozdělena do tří hlavních podkapitol pojednávajících o prozkoumávání, transformaci a úpravě těchto kategorií. Výstupy a výsledky jsou poté porovnány s výsledky a závěry datových vědců, kteří taktéž analyzovali dataset firmy Bosch na stránkách Kaggle.

Tato práce a přiložené programy popisují cyklus zpracování velkých dat a podtrhují důvod, proč se při analýze učinily jednotlivé závěry.

1 Koncept velkých dat

1.1 Velká data obecně

Velká data jsou jedním z nejpopulárnějších výzkumných témat dnešních vědeckých a technologických komunit. Mají ve všech odvětvích naši společnosti velký potenciál. Ať už mluvíme o informačních technologiích, klimatu, financích, hospodářství nebo i zdravotnictví, velká data můžeme najít zkrátka všude.

Problematika velkých dat je stále na svém vzestupu a má před sebou dlouhou cestu, o čemž svědčí nevyřešená bezpečnost, bariéry v soukromí pro všudypřítomnou implementaci a také její vágní definice. Je tedy jisté, že budeme čelit mnoha nebývalým problémům a výzvám při objevování této revoluční kapitoly lidských dějin.

Jelikož velká data vyžadují speciální technologie a metody pro zpracování a analýzu, je nevyhnutelně nutné integrovat do informačních systémů pro velká data znalosti z tradičních databázových systémů, které mají jasně definovaný matematický základ, soubor designových pravidel a implementačních mechanismů (Yu, Guo 2016).

Autoři De Mauro, Marco Greco a Michele Grimaldi (2016) uvádějí, že rozvíjející se disciplíny se často setkávají s nedostatkem shody ohledně definic jejich základních pojmů. Velká data nejsou výjimkou. V literatuře můžeme najít hned několik definic. Jeden z důvodů, proč tomu tak je, je velmi rychlý nástup a vývoj literatury dané problematiky. Je zde tedy úskalí, že nemáme všeobecně a formálně přijímanou definici pro velká data.

Absence dohodnuté definice velkých dat často vedla vědce k přijetí "implicitních" definic prostřednictvím aktuálních trendů, úspěšných projektů, charakteristik, technologických vlastností nebo jejich dopadu na společnost, firmy a obchodní procesy. Stávající definice tedy poskytují velmi odlišné pohledy, a proto se velká data považují za pojem, popisující sociální fenomén, informativní majetek, data sety, technologie úložišť, analytické techniky, procesy a infrastruktury (Mauro, Greco, Grimaldi 2016).

Když se podíváme na nejrůznější definice Velkých Dat v odborné literatuře, můžeme je rozdělit do čtyř skupin podle toho, na co při popisu tohoto fenoménu byl kladen důraz:

- Vlastnosti dat
- Technologické potřeby
- Překonání prahových hodnot
- Sociální dopad

Analýza stávajících definic pro velká data nám umožňuje dospět k závěru, že jádro konceptu velkých dat zahrnuje následující aspekty:

- Objem, Rychlost a Rozmanitost (Volume, Velocity, Variety), sloužící k popisu charakteristiky informací.
- Technologie a Analytické metody, sloužící k popisu nezbytných požadavků pro správné používání těchto informací.
- Hodnota, popisující transformaci informací do poznatků, které mohou znamenat ekonomickou hodnotu pro firmy a společnost.

Definice velkých dat by měla odkazovat na jejich hlavní povahu, to znamená jejich "informační hodnotou". Jedná se o jasně definovanou entitu, která není závislá na oblasti použití. Autoři Andrea de Mauro, Marco Greco a Michele Grimaldi (2016, s.7) navrhují následující definici:

"Velká data jsou informační aktivum, charakterizováno určitým velkým o*bjemem (volume),* r*ychlostí (velocity) a* r*ozmanitostí (variety), které vyžaduje speciální* t*echnologii a* a*nalytické metody pro jeho transformaci na* h*odnotu."* (překlad vlastní)

1.2 Velká Data a Průmysl 4.0

Velká Data jsou v dnešním průmyslu často využívána, jelikož se každým dnem navyšuje množství generovaných dat tímto odvětvím. Díky rychlému vývoji konceptů jako jsou velká data, strojové učení a internet věcí se zrodil nový koncept Průmysl 4.0, který označuje čtvrtou průmyslovou revoluci (Witkowski 2017).

Moderní továrny se stále posouvají a jsou složitější a propojenější než kdy dříve. Vznikají zde však také nové výzvy, kterým může inteligentní automatizace podporována Velkými Daty čelit. Narůstající množství informací z internetu věcí a moderních systémů umožňuje inteligentním továrnám zvyšovat svou efektivitu a výrazně vylepšit životnost spolu se zrychlením výroby a snížením kritických chyb (Nexus Integra 2020).

2 Nástroje a metody pro sběr velkých dat

2.1 Základní koncepty

Sběr dat je proces shromažďování, filtrování a čištění dat před jejich vložením do databáze nebo jakéhokoli jiného úložiště, nad kterým lze provádět analýzu dat. Sběr dat je jednou z hlavních výzev velkých dat z hlediska požadavků na infrastrukturu a první problém který se v životním cyklu velkých dat musí vyřešit. Infrastruktura potřebná k podpoře získávání velkých dat musí poskytovat nízkou, předvídatelnou latenci jak při snímání dat, tak při provádění dotazů; být schopna zpracovat velmi vysoké objemy transakcí, často v distribuovaném prostředí; a podporovat flexibilní a dynamické datové struktury (Cavanilas 2016).

Většina scénářů při sběru velkých dat předpokládá velkoobjemové, vysokorychlostní, velmi rozmanité, ale málo hodnotné údaje. Proto je důležité mít adaptabilní a časově efektivní algoritmus pro sběr, filtrování a čištění dat, který zajistí, že se analýza provede pouze na datech, které mají potenciální hodnotu (Cavanillas 2016).

Když se podíváme na jedny z největších architektur pro zpracovávání velkých dat, jako je například Oracle (Oracle, 2012) nebo IBM (IBM, 2013), můžeme si všimnout dvou společných prvků pro sběr dat:

- Protokoly, které umožní shromáždění informací o zdrojích dat jakéhokoliv typu (strukturované, nestrukturované, polostrukturované).
- Rámce a softwarové nástroje, pomocí kterých jsou data shromažďována ze zdrojů pomocí různých protokolů.

J.M. Cavanillas (2016) dále ještě také uvádí třetí prvek, a to jsou databáze kde se data uchovají po zpracování softwarovými nástroji. Databáze však budou samostatně popsány v kapitole *4. Úložiště pro velká data*.

2.2 Protokoly

2.2.1 Advanced Message Queuing Protocol (AMQP)

Oasis (2012) uvádí, že AMQP je otevřený internetový protokol pro zasílání zpráv, který je definovaný na binární úrovni. Toto umožňuje spolehlivou výměnu zpráv mezi dvěma stranami. Bank of America (2011) jako hlavní charakteristiky AMQP definují všudypřítomnost, bezpečnost, spolehlivost, interoperabilitu, věrnost, použitelnost a ovladatelnost. Tyto charakteristiky pak autor J.M. Cavanillas (2016) dále popisuje.

- **Všudypřítomnost:** tato vlastnost AMQP odkazuje na jeho schopnost využití v různých průmyslových odvětvích v současných i budoucích architekturách pro sběr dat. Tato charakteristika je dosažena velmi jednoduchou implementací. Toto potvrzuje velké

množství rámců, které tento protokol používají: SwiftMQ, Microsoft Windows Azure Service Bus, Apache Qpid a Apache ActiveMQ.

- **Bezpečnost:** tato charakteristika je implementovaná na dvou různých úrovních. První úroveň je integrace šifrování zpráv, díky tomu ani zachycené zprávy třetí stranou není možné jednoduše dekódovat. A druhá úroveň zabezpečení zajišťuje dlouhou trvanlivost zprávy, což znamená, že zprávy je možné odeslat i když odesílatel a příjemce nejsou připojeni k síti ve stejný čas.
- **Věrnost:** tato charakteristika se zabývá integritou zprávy. AMQP poskytuje prostředky k zajištění toho, aby odesilatel mohl vyjádřit strukturu zprávy a umožnit tak příjemci pochopit, co předem může očekávat. Protokol také implementuje spolehlivou detekci selhání, která umožňuje systému detekovat chyby už při vytvoření zprávy na straně odesílatele ještě předtím, než si informaci uloží příjemce.
- **Použitelnost:** záměrem této vlastnosti je zajistit, aby klienti a zprostředkovatelé AMQP mohli komunikovat pomocí několika protokolů vrstev modelu OSI. Jako jsou například TCP, UDP a také SCTP protokoly. Díky tomu je AMQP použitelný v mnoha průmyslových odvětvích, kde nejsou využívány a používány všechny vrstvy modelu OSI.
- **Interoperabilita:** AMPQ je navržený tak aby byl nezávislý na konkrétních implementacích a prodejcích. To znamená, že klienti a zprostředkovatelé s plně nezávislými architekturami můžou spolu komunikovat právě skrz AMPQ.
- **Spravovatelnost:** jedním z hlavních problémů během specifikace AMQP, bylo zajistit škálovatelnost rámců, které jej implementují. Tato charakteristika je naplněna tím, že AMQP je odolný a bezztrátový protokol, přes který lze přenášet informace všech typů (XML, audio, video atd.).

Aby všechny tyto charakteristiky byly správně implementovány AMQP se spoléhá na typový systém a čtyři vrstvy: transportační, transakční, zasílací a bezpečnostní. Typový systém je podle Bank of America (2012) systém definující sadu běžně používaných primitivních typů používaných pro interoperabilní reprezentaci dat.

2.2.2 Java Message Service (JMS)

Oracle (1999) definuje JMS jako běžný způsob, jakým programy Java vytvářejí, odesílají, přijímají a čtou zprávy podnikových systémů. Rozhraní JMS podporuje dva běžně využívané programovací modely: publish-and-subscribe a point-to-point. Každý model poskytuje své výhody poskytovatelům JMS, kteří můžou implementovat jeden nebo oba modely najednou. Systémy JMS poskytuje vývojářům prostředí Java systém asynchronního zasílání zpráv, který umožňuje interakci systémů bez nutnosti jejich propojení. Zprávy lze odesílat do systémů, které momentálně nejsou spuštěny, a zpracovávat je, když je to vhodné a efektivní. Oddělené a asynchronní charakteristiky zpráv činí z JMS výkonné a kritické podnikové API.

JMS je také kompatibilní s dříve zmiňovaným AMQP, což je v podstatě standart pro předávání zpráv ve světě Java. Zatímco AMQP je definován formátem, JMS je standardizován na úrovni API, a proto není snadné jej implementovat v jiných programovacích jazycích. JMS také neposkytuje funkce pro vyvažování zátěže, odolnosti proti chybám, upozornění na dané chyby, správu služeb, zabezpečení a další.

2.3 Softwarové nástroje

2.3.1 Storm

Apache Storm je otevřený distribuovaný výpočetní systém pro výpočty v reálném čase. Apache Storm podporuje širokou nabídku programovacích jazyků a také databází (J.M. Cavanillas 2016).

Storm vyvinul Nathan Marz. Později jej převzala společnost Twitter v roce 2011. Poté v roce 2013 Twitter umístil Storm na GitHub a udělal z něho volně dostupný produkt. Ve stejném roce poté Storm vstoupil do Apache Software Foundation. Od této doby splňuje Apache Storm požadavky na zpracování velkých dat (Intellipaat 2016).

Tiwari 2017 děli architekturu Storm na dva uzly:

- **Hlavní uzel (Nimbus):** hlavní uzel celé architektury Storm, nazývaný také Nimbus je velmi podobný "Job Tracker" v Hadoob. Nimbus je zodpovědný za rozdělení kódu, přiřazování jednotlivých úkolů a monitorování výkonu. Nimbus je služba Apache Thrift, což umožňuje zadávání kódu v libovolném jazyce. Nimbus monitoruje zpracování zpráv skrz Apache Zookeeper, kde všechny pracovní uzly zasílají aktualizace jejich stavu.
- **Pracovní uzly (Supervisors):** Každý pracovní uzel si spustí deamona zvaného Supervisor, který sleduje práci odvedenou strojem, který mu byl přidělen. Supervisor podle potřeby spouští a zastavuje pracovní procesu na základě příkazu Nimbuse.

Dále Tiwari 2017 uvádí, že výpočet v Apache Storm se provádí skrz takzvané Topologie. Topologie je implementovaná jako datová struktura DAG (acyklický orientovaný graf). V tomto grafu reprezentují hrany takzvané proudy (streams) a jednotlivé uzly Bolts nebo Spouts.

- **Proud** (Stream)**:** reprezentuje neomezené sekvence dvojic dat (v datovém modelu klíč-hodnota). Proudy dvojit dat se pohybují mezi jedním uzlem Spout a několika Bolts, nebo jenom mezi několika Bolts. Existuje několik technik, jakými můžeme jednotlivé Proudy definovat.
- **Spouts:** jsou vstupními body celé Topologie Apache Storm. Obecně vždycky Spouts budou číst dvojice dat z nějakého externího zdroje a podávat je dále do Topologie Apache Storm.
- **Bolts:** se starají o veškeré procesy v Topologii Apache Storm, tyto uzly mohou dělat všechno od filtrování po zpracování funkcí až po komunikaci s databázemi.

2.3.2 **Kafka**

Apache Kafka je distribuovaný systém, který se skládá ze serverů a klientů. Klienti a server mezi sebou komunikují prostřednictvím síťového protokoly TCP. Apache Kafka funguje jako soubor jednoho nebo více serverů, které se skládají z jednoho nebo více datových center a cloudů. Kafka dělí servery na několik druhů, jeden z nich je takzvaný zprostředkovatel (Broker). Ostatní servery má na starosti Kafka Connect, tento servis nepřetržitě importuje a exportuje data jako proudy událostí, aby mohl být Apache Kafka jednoduše integrovaný do již existujících systémů (relační databáze nebo jiné Kafka systémy). Klastr Kafka je vysoce škálovatelný a odolný vůči chybám, jestli některý ze serverů selže, tak ostatní servery převezmou jeho práci, aby se zajistila nepřetržitý provoz s žádnými ztráty na datech. Klienti poté umožňují vytvářet distribuované aplikace a mikro služby. Tyto služby a aplikace jsou schopné číst, zapisovat a zpracovávat proudy událostí paralelně i v případech selhání sítě nebo daného stroje (Apache Software Foundation 2017).

Vinka a Johansson (2019) uvádějí a popisují několik základních konceptů z kterých se architektura Apache Kafka skláda: záznamy, témata, producenty, spotřebitele, oddíly témat (nebo také protokol o potvrzení) a posun oddílu.

Veškeré záznamy (records) v Apache Kafka jsou rozděleny do témat (topics). Témata jsou kategorie v brokeru Apache Kafka, kde jsou záznamy publikovány. Jednotlivé data v záznamu se můžou ukládat v různých datových typech, jako je například řetězec znaků nebo také složitější datové struktury jako jsou JSON. Záznamy do jednotlivých témat vkládá producent (Producer) a odběr daného tématu spotřebitel (Consumer). Apache Kafka umožňuje spotřebovávat záznamy různým tempem v závislosti na jejich konfiguraci. Dané záznamy poté zůstanou v tématu tak dlouho dokud nebude překročen časový limit jejich uchování nebo velikost záznamu.

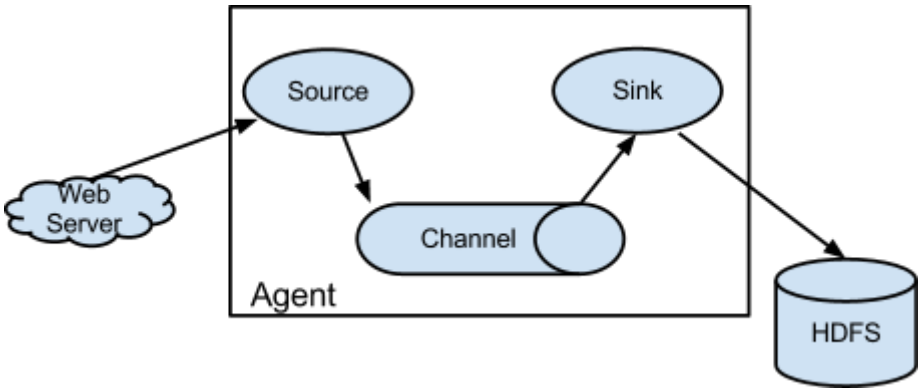
Jednotlivé téma jsou rozdělené do oddílů, které uchovávají záznamy v nezaměnitelném pořadí. Oddíl se může také někdy nazývat Protokol Potvrzení. Oddíly umožňují paralelizovat jednotlivá témata tím, že rozdělení data jednoho téma na více uzlů. Každý záznam v oddílu je identifikován jedinečným posunem, tento posun ukazuje na záznam v daném oddílu (příchozí nové záznamy jsou poté připojeny na konec oddílu jako ve frontě). Jednotlivé témata můžou mít také více oddílů, aby se umožnilo spotřebitelům paralelní čtení z témat.

2.3.3 **Flume**

Flume je jeden z dalších Apache distribuovaných systémů pro efektivní sběr, agregaci a přenos velkého množství dat z různých zdrojů do jedné centralizované databáze (Apache Software Foundation 2019).

Z knihy od autora Steve Hoffmana (2015) můžeme architekturu Apache Flume rozdělit na několik částí jako zdroje dat (sources), klienty (clients), tok dat (data flow), události (events), kanály (channels), agenti (agents) a odtoky dat (data Sinks).

Vstup Agenta se nazývá zdroj, který zapisuje události dat do jednoho nebo více kanálů a výstupy zvané odtoky naopak zase čtou datové události z jednoho nebo více kanálů viz. obrázek 2.1 níže. Kanály jsou spojovací prostředek mezi zdroji a odtoky, které v sobě ukládají datové události. Agent se tedy skládá z jednoho zdroje, odtoku a jednoho nebo několika kanálů (kdy v daných kanálech se může nacházet několik datových událostí). Událost je tedy základní datový kontejner pro přenos dat v Apache Flume, který se skládá z žádné nebo více hlaviček a jednoho těla. Hlavičky jsou páry datového modelu klíč-hodnota, které se využívají pro rozhodování o směru dané události nebo k přenosu dalších užitečných informací jako jsou například časová razítka nebo název hostujícího serveru (funkci má podobnou hlavičce v HTTP). Tělo potom nese hodnotnou informaci.



Obrázek 2.1:"*Tok dat v Apache Flume*"

Tok dat v Apache Flume začíná u Klienta (Client) (viz. obrázek výše). Klient přenáší generované události do další destinace. Tato destinace je zdroj dat (který se však nachází v agentovi). Zdroj přijímající tuto událost jí předá jednomu nebo více kanálů, které tuto událost uchovají a dále ji předají odtoku. Pokud se jedná o základní odtok, kterému jsou události předána, předá událost dalšímu agentu. Pokud se však jedná o takzvaný odtok terminálu (terminal sink), tak se přepošle událost už do konečného cíle. Tento model předávání dat mezi zdroji a odtoky je známy jako producer-consumer, který umožňuje, aby zdroje a odtoky dat měli různé charakteristiky výkonu a prostředí na kterém běží, a přesto mohly efektivně využívat dostupné fyzické prostředky (J.M. Cavanillas 2016).

2.3.4 **Hadoop**

Hadoop je systém dávkového zpracování pro síť výpočetních uzlů, který také poskytuje základ většiny analytických prvků pro Velká Data. Toto je možné díky sady dvou funkcí, které jsou nejvíce potřebné pro řešení velkých nestrukturovaných datových sad: MapReduce a systém distribuovaných souborů (HDFS). Hadoop je také projekt od Apache Software Foundation, který je napsaný v Javě pro podporu datově náročných distribuovaných aplikací. Hadoop umožňuje aplikacím pracovat s tisíci uzly a daty s velikostí dosahující petabajtů. Inspirací pro Hadoop vychází z dokumentů MapReduce a Google File Systém od společnosti Google (Padhy 2013).

Prajapati (2013) popisuje HDFS jako vlastní souborový systém od Hadoop. Jednou z významných charakteristik Hadoopu je rozdělení a výpočet napříč až tisíci hostiteli a paralelní výpočty aplikací v blízkosti zpracovaných dat. Na HDFS se datové soubory kopírují jako sekvence bloků v klastru. Klastr Hadoop je velmi škálovatelný a dokáže přizpůsobit svou výpočetní kapacitu, velikost databází a šířku pásma vstupů a výstupů pouhým přidáním komoditních serverů. K HDFS lze z aplikací přistupovat mnoha různými způsoby (v základu HDFS poskytuje rozhraní JAVA API pro použití). Další důležité charakteristiky HDFS jsou tolerance chyb, schopnost fungovat na komoditním hardwaru, schopnost zpracovávat velké datové sady, master-slave paradigma a přístup k souborům zvaný write-once-file-acces-only (Prajapati 2013).

Dále autor Prajapati (2013) také mluví o MapReduce, který popisuje jako programovací model pro zpracování velkých datových sad distribuovaných ve velkém klastru. Dalo by se říct, že mapReduce je srdcem Hadoop. Programovací paradigma MapReduce umožňuje provádět masivní zpracování dat napříč tisící serverů nakonfigurovaných s klastry systému Hadoop. MapReduce je odvozený od Google MapReduce. Hadoop je softwarový rámec pro snané saní aplikací, které spolehlivě a bezchybně dokáze zpracovávat i velké množství dat paralelně na velkých klastrech komoditního hardwaru (třeba až tisíce uzlů). Paradigma MapReduce je rozděleno do dvou fází, Map a Reduce (výstup fáze Map je vstup pro fázi Reduce).

Tyto dvě fáze můžeme vysvětlit následovně (Tutorials Point 2020):

- Mapovací fáze (Map): Jakmile jsou data rozdělena, jsou sledovačům úkolů přiřazeny datové sady, aby nad nimi provedly fázi mapování. Datová funkční operace bude provedena nad daty, přičemž jako výstup fáze Map bude pár mapovaných klíčů a hodnot (data s modelem key-value).
- Redukovací fáze (Reduce): Hlavní uzel poté shromáždí data (odpovědi) ze všech dílčích problémů a nějakým způsobem je nakombinuje, aby vytvořil výstup. Tento výstup by měl odpovídat na problém, který se původně celá operace snažila vyřešit.

3 Nástroje pro analýzu velkých dat

3.1 Základní koncepty

Analýza velkých dat spojuje tradiční přístupy analýzy statistických dat s těmi výpočetními. Statistická analýza vzorkování je vhodná, když je k dispozici celá datová sada. Tato podmínka je celkem typická pro tradiční scénáře dávkového zpracování. Velká Data však mohou posunout dávkové zpracování na zpracování dat v reálném čase kvůli časté potřebě dát smysl datovým proudům, které zpracováváme. U streamovaných dat se datová sada datová sada pomalu hromadí a data jsou například s časovým razítkem seřazená. Streamování dat klade důraz na rychlé zpracování, jelikož analytické výsledky mají někdy dočasnou hodnotu.

V každém rychle se rozvíjejícím oboru, nevyjímaje Velká Data, existuje řada příležitostí pro inovace. Příkladem tohoto jevu je otázka, jak nejlépe spojit statistické a výpočetní přístupy pro daný analytický systém. Přechod od dávkového zpracovávání na analýzu v reálném čase je také výzva, se kterou nástroje pro analýzu Velkých Dat musejí počítat (Erl, Khattak, Buhler 2015).

3.2 Softwarové nástroje

3.2.1 Weka

Cílem projektu Weka je poskytnout uživateli široké spektrum nejmodernějších algoritmů pro strojové učení a nástrojů pro předběžné zpracování dat. Weka umožňuje rychle vyzkoušet a porovnávat různé algoritmy strojového učení nad různými datovými sadami. Jeden z ryzích charakteristik Weka je modularita a rozšiřitelná architektura, která umožňuje vybudování složitých procesů pro sběr dat z širokého spektra poskytovaných algoritmů a nástrojů. Díky snadno pochopitelnému API a pluginům, které automatizují integraci nových algoritmů do Weka, je rozšíření sady nástrojů velmi jednoduché. Weka nabízí algoritmy pro regresi, klasifikaci, dolování asociačních pravidel, shlukování a výběr atributů (Witten et al. 2009).

Aby byl provoz Weky co nejflexibilnější, byl systém Weka navržen s modulární objektově orientovanou architekturou, která umožňuje snadné přidávání nových nástrojů, algoritmů, modelů, filtrů atd. Pro každý hlavní typ komponent byla vytvořena abstraktní třída v Javě pro zajištění ještě jednodušší implementace.

Všechny klasifikátory jsou uloženy a umístěny v dílčích složkách balíčků klasifikátoři (Classifiers), který je nejvyšší úrovně a rozšiřují základní třídu s názvem klasifikátor (Classifier). Klasifikátoři jsou organizovaní podle obecného typu algoritmu, který využívají, takže najdeme balíčky třeba pro Bayesovské metody, rozhodovací stromy atd.

Všechny komponenty Weka se spoléhají na podporu tří, které jsou umístěny balíčku zvaném jádro (core). Jádro poskytuje datové struktury a třídy, které čtou datové sady, představují instance, vlastnosti a poskytují různé základní metody obslužných programů. Základní balíček

také obsahuje rozhraní, které mohou jednotlivé komponenty implementovat, aby bylo jasné, že dané funkce podporují. Avšak jeden z hlavních částí Weka pro koncového uživatele je grafické rozhraní. Aby se zachovala modularita muselo se využít introspekčních mechanismů Javy, které umožní dynamicky konfigurovat nastavení jednotlivých komponent za běhu aplikace (Frank et al 2009).

Dean Jared (2014) varuje, že Weka není ale dobře škálovatelná, jelikož je limitovaná zdrojem RAM, typicky jednoho počítače. Proto dokumentace Weka nasměřuje uživatele, aby se data před samotnou analýzou řádně předzpracovala, očistila a filtrovala.

3.2.2 **Java**

Programovací jazyk Java je jeden z nejvíce používaných jazyků v kontextu analýzy a zpracování velkých dat. Je tomu zejména kvůli tomu, že většina nástrojů (Hadoop, Storm, Spark atd.) jsou napsány jako open-source projekt v Javě. To znamená, že je umožněné komukoliv tyto nástroje využívat a také díky tomu jsou hodně využívány v širokém spektru IT společností (CodeGym 2020).

Dean (2014) uvádí dva hlavní důvody proč Java nebyla zpočátku moc populární jako jazyk pro analýzu velkých dat. Základní knihovna pro numerické výpočty byla pomalejší než u jiných jazyků a JVM značně zpomalovala běh aplikace díky správě paměti. Dnes se ale tyto problémy přehlížejí, jelikož dramaticky vzrostla složitost aplikací pro analýzu velkých dat a současně s tím také náročnost vývoje, a tudíž se stal čas na vývoj aplikací dražší, než samotné nároky na CPU a výkon hardwaru.

Z mnoha knihoven, které jsou volně přístupné pro vývoj aplikací na analýzu velkých dat si popíšeme hlavně Spark MLlib a DeepLearning4j pro jejich popularitu v Java komunitě (Zvehlev, Rozeva 2018):

- **MLlib:** je knihovna pro strojové učení z rámce Apache Spark. Tato knihovna poskytne uživateli základní a běžné algoritmy pro strojové učení jako je klasifikace, regrese, shlukování a filtrace. Apache Spark (2018) na svých stránkách přímo uvádějí, že některé algoritmy jsou až 100x rychlejší než jejich obdoba v prostředí Hadoop.
- **DeepLearning4j:** je obsáhlá Java knihovna pro hloubkové učení. Tato knihovna je open-source, a tudíž u ní můžeme najít velkou a aktivní komunit. DeepLearning4j nabízí širokou škálu hloubkové učení, které se dají využít například pro analýzu obrazu, prediktivní analytika pro proudy velkých dat a detekce anomálií v systémech.

3.2.3 **R**

R je open-source programovací jazyk čtvrté generace určený pro statistickou analýzu. R má základy v komerčním jazyce S, který byl vyvinut koncem 70. let v Bell Laboratories. R má širokou a neustále rostoucí komunitu a vybudoval si své místo mezi akademické pracovníky zabývajícími se datovou vědou a analýzou velkých dat. Je také známo, že velké společnosti jako je například

Bank of America, Google a Shell využívají programovací jazyk R pro nejrůznější účely a aplikace (Dean 2014).

Programovací jazyk R poskytuje vývojářům více než 3000 různých balíčků a tento počet každým dnem roste Prajapati (2013).

Balíček je navzájem související sada funkcí, soubor s nápovědami a datových soubor, které byly spojeny dohromady. Balíček R mají podobnou funkci jako moduly v Perlu, knihovny v C/C++ a třídám v programovacím jazyce Java. Všechny funkce v Balíčku mají většinou funkčnost zaměřenou stejným směrem, takže například v balíčku pro statistiku najdeme funkce pro provádění statistickou analýzu. R nabízí enormní množství nejrůznějších balíčků počínaje od balíčků zobrazující nějakou grafiku, provádění statistických testů a balíčky pro nejnovější metody a algoritmy strojového učení. Můžeme v R najít také balíčky pro velké spektrum průmyslových odvětví a aplikací. Také u R se základní stavba programu skládá ze základních operací (jako jsou například aritmetické sčítání či bitové operace), deklarování funkcí a jejich volání, proměnných, složitějších datových struktur, objektů a tříd a také takzvaných modelů, které popisují kus nějakých dat matematickou rovnicí (Adler 2010).

Když čelíme nějaké úloze, které je výpočetně náročná prostředí R umožňuje volat kód naprogramovaný v jiných programovacích jazycích (jako je například C / C++ nebo Firtran). Pro zkušenější vývojáře je možnost volat objekty R v jazyce C (Chen, Mao a Liu 2014).

Autor John Dean (2014) popisuje a vyjmenovává jedny z nejpopulárnějších balíčků R:

- **Snow**: se využívá pro spouštění výpočtů R na klastru několika počítačů najednou. Ke komunikaci mezi jednotlivými uzly Snow využívá například sockety nebo NetWorkSpaces. Tento balíček se využívá převážně na náročné simulace a využívá tradiční paralelizační architekturu master-worker.
- **Multicore:** je jednoduchý balíček, který rozděluje jedno vláknové sekvenční procesy R do procesů běžícím na stejném počítači ale na více vláknech.
- **Rhadoop a Rhipe:** balíčky umožňují vývojáři programový přístup k systému Hadoop a Hipe pomocí jazyka R. Vývojář tedy může využívat MapReduce funkce na Hadoop klastru skrz programovací jazyk R.

3.2.4 **Python**

Python se stal lingua franca pro mnoho aplikací a systémů datové vědy a zpracování velkých dat. Tento jazyk kombinuje sílu univerzálních programovacích jazyků se snadným použitím skriptovacích jazyků pro určitou doménu jako je například Matlab nebo R. Python disponuje ohromným množstvím nejrůznějších knihoven počínajíc od knihoven pro načítání dat, jejich vizualizaci, statistiku, zpracování mluveného jazyka, zpracování obrazů, tvoření aplikací a spousty dalších. Tato rozsáhla sada umožňuje vývojářům. Jednou z hlavních výhod

programovacího jazyka Python je možnost komunikovat s kódem skrz terminál nebo jiné nástroje jako je například Jupyter Notebook (Müller, Guido 2016).

Programy v jazyce Python také mohou volat kompilované nativní binární soubory programovacích jazyků C / C++ nebo také Fortran prostřednictvím několika dostupných API. Python podporuje programy, které mohou běžet na více vláknech a podobně jako jazyky založené na JVM platforem může Python spravovat paměť a elegantně zpracovávat výjimky (Python byl přenesen do platformy JVM v projektu zvaném Jython). Díky tedy všem těmto výhodám v kombinaci s mnoha databázovými, grafickými a matematickými knihovnami je Python pro analýzu velkých dat, jejich těžbu a také na řadu dalších aplikací. Matematické a statistické modely lze implementovat pomocí volně dostupných knihoven Scipy a Numpy a vizualizaci pomocí novější knihovny Matplotlib (Dean 2014).

V krátkosti popíšu některé základní knihovny a nástroje pro práci s velkými daty v programovacím jazyce Python:

- **JupyterNotebook:** je webová interaktivní aplikace, která vývojářům umožňuje vytvářet a sdílet dokumenty s živým kódem, rovnicemi, vizualizací a textem. Využití JupyterNotebooku je široké, od transformací dat po statistické modelování až po strojové učení (Project Jupyter 2020).
- **NumPy:** knihovna poskytuje vývojářům sílu jazyků C a Fortran s jednoduchostí použití programovacího jazyka Python, který toto postrádá (Numpy 2020). Numpy obsahuje funkcionalitu pro multidimenzionální pole, matematické rovnice vyšších úrovní a pseudonáhodné generátory čísel (Müller, Guido 2016).
- **MatPlotLib:** Matplotlib je 2D grafický balíček používaný pro Python pro vývoj aplikací, interaktivní skriptování a generování obrázků v publikační kvalitě napříč uživatelskými rozhraními a operačními systémy (Hunter 2007).
- **Pandas:** Je rychlý, výkonný, flexibilní a snadno použitelný nástroj pro analýzu a manipulaci dat, který je postavený na programovacím jazyce Python (Tým vývojářů Pandas 2020).

4 Úložiště pro velká data

4.1 Základní koncepty

Ukládání velkých dat je srdcem nástrojů a platforem pro práci s velkými daty. Hlavním důvodem je, že pokud velká data různých formátu správně neuložíme, načítání dat a následující výpočty a práce s daty budou velmi neefektivní a pomalé. Hlavní rozdíly v typech úložišť pro velká data se točí kolem základních konceptů, jako jsou modely dat, dělení dat, replikace dat, formát dat, indexování dat a vytrvalost dat (Yu, Guo 2016).

Ideální typ uložiště pro velká data popisují Martin Strohbach, Jörg Daubert, Herman Ravkin, and Mario Lischka (2016) jako systém, který umožňuje ukládání prakticky neomezeného množství dat, který by se rychlostí čtení a zápisu vyrovnal databázím s náhodným přístupem, flexibilně a efektivně by řešil řadu různých datových modelů, podporoval strukturovaná i nestrukturovaná data a z důvodů ochrany osobních údajů pracuje pouze na šifrovaných datech. Je zřejmé, že všechny tyto potřeby nelze úplně splnit. Ale v posledních letech se objevilo spoustu nových úložných systémů a technologií, které tyto problémy řeší alespoň částečně.

4.1.1 Datové modely

Technologie pro práci s velkými daty typicky podporují různé typy data modelů pro reprezentaci dat. Jeden z data modelů, který je široce používaný v průmyslu posledních několik desetiletí, je relační model. Databázové systémy s relačním modelem pro definování a přístup k datům využívají Structured Query Language (SQL) (Yu, Guo 2016).

Existují však také technologie, které nepodporují relační model. Jsou známé jako NoSQL databáze. Tyto databáze podporují datový model založený na klíči / hodnotě. K danému klíči v tomto datovém modelu může být přiřazená jakákoliv hodnota, počínaje od jakéhokoliv primitivního typu (jako celé číslo, znak. Bajt atd.)), přes vícerozměrné mapy, až po složitější struktury jako je například XML nebo JSON.

Dalším zajímavým datovým modelem je model graf, kde jsou data reprezentovaná jako uzly, odkazy, nebo hrany. Datový model graf může však oproti dříve zmíněným datovým modelům představovat jisté problémy (např. výpočet vzdáleností, hledání kružnic ve vztazích, určování konektivity atd.). I přesto má tento datový model v mnoha reálných případech své použití, mnohdy související se sociálními médii, sítích finančních transakcí, sítí zákazníků atd.

Bez ohledu na modely, je většina technologií vyvinuta na základě konceptu "definice modelu dat v reálném čase", který je také známy jako "scheme on read". Tento přístup je ve srovnání s přístupem používaným v tradičních databázových technologiích, kde je třeba před přesunem dat na platformu definovat jejich schéma, velmi efektivní.

4.1.2 **Dělení dat**

Každá technologie velkých dat musí dodržovat určitý přístup k rozdělení dat mezi různé datové uzly. Důvodem je, že všechna data nelze uložit na jednom počítači, a přitom také zajistit, aby zpracování dat mohlo probíhat paralelně s využitím všech dostupných výpočetních prostředků. U dat modelovaných a přístupných ve formě klíče / hodnoty nebo klíče / n-tice se se dělení provádí na základě klíče. K dosažení požadovaného rozdělení lze použít různé typy schémat rozdělení: rozdělení na oblasti pomocí rozsahu hodnot, rozdělení na základě hash funkce, rozdělení na seznam, náhodné rozdělení, pomocí mechanismu Round Robin a rozdělení na základě uživatelem určených značek.

U dat uložených a hromadně přistupovaných bez jakéhokoli předem definovaného schématu se používá blokový přístup k rozdělení oddílů. V tomto přístupu je každý po sobě jdoucí blok bajtů, založený na konkrétně nakonfigurované velikosti bloku, zapsán do různých uzlů.

Pro rozdělení dat grafu se obvykle používají dva přístupy – řez vrcholem nebo řez hranou v závislosti na řešeném problému jaký mají data popisovat. Některé algoritmy zpracování také používají hybridní přístup.

4.1.3 **Formát dat**

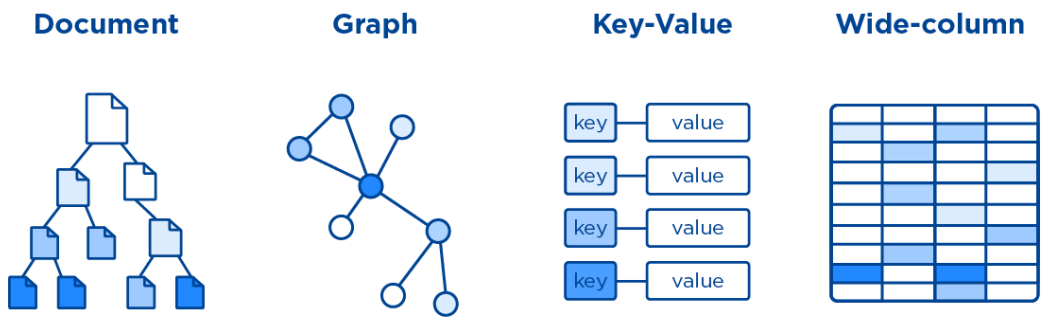
Formát dat databází používaných k ukládání a zpracování dat v technologiích pro Velká Data vyžaduje speciální pozornost, jelikož je to jeden z hlavních aspektů ovlivňujících škálovatelnost a flexibilitu technologií pro Velká Data. Níže jsou popsány populární formáty:

- Textové soubory s oddělovači: Jedná se o jednoduché soubory, kde jsou data ukládána v lidském čitelném formátu. Záznamy jsou zde odděleny znakem nového řádku '\n' a pole jsou běžně oddělena čárkou nebo například tabulátorem. Příklady populárních typů souborů s oddělovači jsou CSV (hodnoty oddělené čárkami) nebo také TSV (hodnoty oddělené tabulátorem)
- Parquet: jsou sloupcová úložiště kde jsou místo všech hodnot řádku hodnoty sloupce a naopak. Parquet byl postaven od základu pro podporu složitě větvených dat pomocí algoritmu skartace a sestavení záznamu, Dremel namísto zploštění větvených jmenných prostorů komplexních dat. Parquet může být použit pro jakýkoliv datový rámec pro zlepšení čtení, zápisu a zpracování dat.
- Optimalizované sloupcové soubory řádků: jedná se také o formát sloupcových úložišť s filozofií ukládání všech hodnot sloupce společně namísto všech hodnot řádku. Tento formát ukládá data do pruhů a uchovává další informace (jako index, agregáty atd.) v datových blocích, kde jsou pruhy uloženy. Tento datový formát je široce využívaný technologiemi Hive.
- Avro: je formát ukládající data po řádcích namísto po sloupcích jako například Parquet. Data v Avro jsou uloženy v souborech spolu s daným schématem, aby se zajistila kompatibilita pro různé programy, které by data v budoucnu chtěli zpracovat.

- Sekvenční soubory: jsou souboru skládající se z dvojic klíč – hodnota. Existují celkem tři typy formátů sekvenčních souborů: nekomprimované, soubory, kde jsou komprimované pouze hodnoty, a soubory kde jsou komprimované jak hodnoty, tak klíče. Sekvenční soubory se značně používají k ukládání dat, který se obtížně dělí jako například XML nebo JSON.

4.2 NoSQL databáze

NoSQL databáze jsou navrženy pro škálovatelnost, často za cenu obětování konzistence dat. Ve srovnání s relačními databázemi často využívají NoSQL databáze rozhraní, které je nízko úrovňové a nestandardizované, což ztěžuje jejich integraci do existujících aplikací, které očekávají rozhraní pro SQL. Databáze NoSQL lze odlišit podle datových modelů, které využívají (viz obrázek níže) (Cavanillas, Curry, Wahlster 2016).



Obrázek 4.1:"*Typy NoSQL databází*"

Úložiště s datovým modelem klíč – hodnota umožňují ukládání dat bez schématu. Datové objekty mohou být zcela nestrukturované nebo strukturované, a jsou přístupné jediným klíčem. Protože tento druh databáze nevyužívá žádné schéma, není ani nutné, aby datové objekty sdílely stejnou strukturu.

Sloupcová úložiště se obvykle skládají z rozptýlených vícerozměrných map, ve kterých jsou data indexována pomocí trojice skládající se z klíče sloupce, klíče řádku a časového razítka. V databázi je hodnota uchovávána jako nepřetržitý řetězec znaků.

Databáze dokumentů na rozdíl od úložišť s datovým modelem klíč – hodnota jsou strukturované. Neexistuje však požadavek na jedno společné schéma, které musí dodržovat všechny dokumenty, jako například u záznamů v relační databázi. Podobně jako u databází klíč-hodnota je možné dokumenty vyhledávat pomocí jedinečného klíče, k datům je však možné přistupovat také skrz dotaz na strukturu požadovaného dokumentu. Databáze dokumentů se tedy označují jako databáze ukládající polostrukturovaná data (Cavanillas, Curry, Wahlster 2016).

A nakonec databáze grafů ukládá data do grafové struktury, které je činí velmi vhodnými kandidáty pro ukládání vysoce asociativních dat, jako jsou například data čerpána ze sociálních sítí. Existuje také speciální typ této databáze, která se zaměřuje na ukládání RDF trojic. Tyto technologie však ještě nejsou vhodné a efektivní pro ukládání skutečně velkých datových sad (Cavanillas, Curry, Wahlster 2016).

4.3 NewSQL databáze

NewSQL databáze jsou moderní formou relačních databází, jejichž cílem je srovnatelná škálovatelnost s NoSQL databázemi zároveň se zachováním transakčních záruk, jaké tradiční databázové systémy nabízejí**.** Podle Venkatesh a Nirmala (2012) mají následující charaktery:

- SQL jako primární mechanismus pro interakci aplikace
- ACID podpora transakcí
- Neuzamykatelný mechanismus pro řízení souběžnosti
- Architektura, která poskytuje mnohem vyšší výkon na jednotlivé datové uzly
- Škálovatelná architektura bez sdílení, schopná provozu na velkém počtu uzlů, bez rizika zpomalování zbytku systému (bottleneck efekt).

Očekává se, že NewSQL databáze jsou přibližně 50krát rychlejší než tradiční OLTP relační databáze.

4.4 Big Data Query platformy

Platformy pro Big Data Query poskytují fasády dotazů nad základními velkými datovými úložišti, což zjednodušuje dotazování na základní datová úložiště. Tyto platformy převážně nabízejí rozhraní podobné SQL pro dotazování a přístup k datům, ale liší se svým přístupem a výkonem.

Hive (Thusoo et al. 2009) poskytuje abstrakci nad Hadoop Distributed File System (HDFS), která umožňuje dotazování strukturovaných souborů pomocí dotazovacího jazyka podobného SQL. Hive provádí dotazy překladem dotazů využitím MapReduce. Mezi výhody Hive patří podobné dotazovací rozhraní jako u SQL a flexibilita pro snadný vývoj schémat. Toto je možné díky tomu, že schéma je uloženo nezávisle na datech a data jsou ověřena pouze během dotazování se na ně. Tento přístup se označuje jako schema-on-read ve srovnání s přístupem schema-on-write v databázích SQL.

Na rozdíl od Hive je Impala (Russel 2013) navržen pro provádění dotazů s nízkou latencí. Znovu používá stejná metadata a uživatelské rozhraní podobné SQL jako Hive, přičemž ale používá vlastní distribuovaný dotazovací engine, který dosahuje nižších latencí.

Spark SQL (Shenker et al. 2013) je další fasáda dotazů s nízkou latencí, která podporuje Hive rozhraní. Autoři projektu tvrdí, že „může provádět dotazy Hive SQL až stokrát rychleji než Hive

bez jakékoli úpravy stávajících dat nebo dotazů“ (Shenker et al. 2013). Tohoto je dosaženo spuštěním dotazů za pomocí rámce Spark (Zaharia et al. 2010) a nikoliv Hadoop MapReduce.

A konečně, Drill je implementace open source aplikace Google Dremel (Melnik et al. 2002), která je podobně jako Impala navržena jako škálovatelný interaktivní ad-hoc dotazovací systém pro vnořená data. Tento jazyk je ale navržen tak, aby podporoval i další dotazovací jazyky, jako je například Mongo Query Language. Na rozdíl od Hive a Impala podporuje Drill řadu datových zdrojů bez schémat, jako jsou HDFS, HBase, Cassandra, MongoDB a SQL databáze.

4.5 Cloud uložiště

Jak roste cloud computing v popularitě, roste také jeho vliv na velká data. Cloud obecně a zejména cloudové úložiště mohou používat jak podniky, tak koncoví uživatelé. Pro koncové uživatelé, znamená využívání cloudu okamžitý přístup, odkudkoliv a z jakéhokoliv zařízení k jejich datům. Koncoví uživatelé mohou navíc využívat cloudové úložiště jako jednoduché pro online zálohu dat z jejich počítače.

Technicky lze cloudové databáze rozlišovat na blokové a objektové. Paměť objektů „je obecný pojem, který popisuje přístup k adresování a manipulaci s diskrétními jednotkami paměti zvanými objekty“ (Margaret Rouse 2014a). Naproti tomu jsou data úložiště bloků ukládána do svazků označovaných také jako bloky. Podle Margaret Rouse (2014b) „každý blok funguje jako samostatný pevný disk“ a umožňuje náhodný přístup k bitům a kouskům dat, což je velmi vhodné pro databáze. Kromě úložišť typu objektů a bloků poskytují hlavní platformy podporu také pro relační a nerelační databázové úložiště, stejně jako pro úložiště typu fronta.

V cloudových úložištích však existuji značné rozdíly, které je nutné brát v potaz při plánování dané aplikace:

- Vzhledem k tomu, že cloudové úložiště je služba, aplikace využívající tyto databáze mají nad nimi menší kontrolu. Také může dojít ke snížení výkonu důsledkem problémů v síti, které nemůžou aplikace nijak ovlivnit. Tyto výkonnostní rozdíly je třeba brát v úvahu při plánování celé aplikace.
- Zabezpečení je jedním z hlavních problémů, s kterými se Cloud databáze potýkají.
- Cloudy s bohatými funkcemi, jako je například AWS, podporují kalibraci latence, redundance a úroveň propustnosti dat, což uživatelům umožňuje najít kompromis mezi cenou a kvalitou.

5 Analýza velkých dat pomocí strojového učení

5.1 Učení s učitelem

Učení s učitelem nebo také prediktivní učení lze rozdělit na dvě kategorie: klasifikace a regrese. Pokud se bavíme o hlavním atributu, který je diskrétní, říkáme, že toto učení je klasifikační. Naopak když je však hlavní atribut spojitý, bavíme se o učení regresí (Wang, Alexander 2016).

Prediktivní učení je jednou z nejčastějších úloh, co se týká těžby dat. Z názvu lze odvodit, že se jedná o proces získávání dat a identifikace vzorů v nasbíraných datech. Tyto vzory lze nalézt skrz modelování a poté využitím daného modelu tvořit predikce na nových vstupních datech (Dean 2014).

5.1.1 Rozdíl mezi učením klasifikací a regresí

V učení klasifikací je cílem předpovědět třídní atribut ze seznamu předem definovaných možností. Zvláštní případ klasifikace je binární klasifikace pro rozlišení dvou tříd. Víceúrovňová klasifikace se snaží vybrat výsledek ze seznamu tříd. Binární klasifikace se dá zjednodušit na příkladu, kdy se snažíme odpovědět na otázku pouhým ano nebo ne (například určení, zdali je email spam nebo nikoliv). Můžeme někdy mluvit také o tom, že jednu třídu považujeme za negativní a druhou za pozitivní (Müller, Guido 2016).

U strojového učení regresí je úkol předpovědět spojitá nebo reálná čísla. Jeden z prostých regresních úkolů je třeba předpovědět příjem člověka z údajů jako vzdělání dané osoby, věku a místa bydlení. V tomto případě předpovídáme hodnotu, která může být libovolné číslo – jedná se tedy o regresi (Müller, Guido 2016).

5.1.2 Lineární regrese

Lineární regrese je jeden z nejzákladnějších modelů pro analýzu velkých dat a strojové učení. Tento model se využívá pro k modelování vztahu mezi několika vstupními a výstupními proměnnými a vrátí nám spojitý výsledek. Klíčovým předpokladem pro tento model je linearita vztahu mezi vstupní a výstupní proměnou nebo proměnnými. I když se tato podmínka může zdát omezující je ve většině případů velmi jednoduché docílit lineárního vztahu mezi vstupními a výstupními proměnnými. Lineární regresní model je tedy takzvaný pravděpodobnostní model, který zohledňuje šum, který může, jakkoliv ovlivnit výsledek. Tento model tedy dokáže s určitou nejistotou předpovědět výstupní proměnné na základě vstupních hodnot (EMC Education Service 2015).

Obecně známá rovnice pro lineární regresní model vypadá následovně (Müller, Guido 2016):

$$\hat{y} = w[0] * x[0] + w[1] * x[1] + \ldots + w[p] * x[p] + b$$

Zde x [0] až x [p] označuje různé vlastnosti jednoho datového bodu (v tomto případě je maximální počet vlastností určen proměnnou p). Parametry modelu jsou w a b, které se učí a ŷ je výsledek (předpověď) modelu. Mezi základní lineární regresi můžeme považovat modely typu lineární regrese (metoda nejmenších čtverců), laso, hřebenová regrese (známa také jako Tikhonova nebo ridge) a lineární modely pro normální a multilevelovou klasifikaci (Müller, Guido 2016).

5.1.3 **Hřebenová regrese**

Hřebenová regrese využívá stejný vzorec pro předpověď hodnot jako typická lineární regrese (metodu nejmenších čtverců). U tohoto modelu jsou však koeficienty "w" voleny tak aby nejen předpovídaly hodnotu na trénovacích datech, ale také aby vyhovovaly nějakému dalšímu omezení: chceme, aby velikost všech koeficientu w se blížila nule. To pro náš model bude znamenat, že každá vlastnost modelu by měla mít co nejmenší vliv na výsledek, a přitom se stále dobře předpovídala. Tento druh omezení spadá pod příklady, které nazýváme regularizace.

Model dělá kompromis mezi jednoduchostí modelu (téměř nulové koeficienty) a jeho výkonem v tréninkové sadě. Jak velký význam model klade na jednoduchost versus výkon tréninkové sady, může uživatel většinou určit pomocí parametru alfa (Müller, Guido 2016).

5.1.4 **Laso**

Alternativou k Hřebenové lineární regresi je model nazývány Laso. Stejně jako u předchozího typu regrese tak také Laso omezuje koeficienty ale odlišnou metodou, tzv. regularizace. Tato metoda některé koeficienty kompletně ignoruje a nastaví jejich hodnotu na nulu. Tuto formu lze považovat za automatické vybírání funkcí a může se využívat pro snadnější interpretaci modelu a odhalení jeho nejdůležitějších funkcí.

V praxi se většinou při volbě mezi hřebenovou a Laso regresí volí hřebenová. Pokud však se v datasetu nachází velké množství vlastností nebo atributů a očekáváte, že důležité bude pouze část nebo jen několik z nich, může Laso představovat pro daný dataset lepší a spolehlivější volbu (Müller, Guido 2016).

5.1.5 **Lineární regrese pro normální a víceúrovňovou klasifikaci**

Lineární modely jsou také využívány pro klasifikaci. Jeden z typů je lineární regrese pro normální klasifikování. Obecná rovnice vypadá takto (Müller, Guido 2016):

$$\hat{y} = w[0] * x[0] + w[1] * x[1] + \ldots + w[p] * x[p] + b > 0$$

Vzorec vypadá velmi podobně jako ten pro lineární regresi, ale namísto vrácení součtu všech funkcí, tak změníme předpovídanou hodnotu na základě podmínky. V našem případě, pokud je funkce menší než nula, předpovídáme třídu -1 v opačném případě předpovídáme +1. Toto pravidlo je společné pro všechny lineární modely pro klasifikaci. Opět existuje mnoho různých

způsobů, jak najít koeficienty (w) a intercept (b). U lineárních modelů pro regresi je výstup ŷ lineární funkcí prvků: přímka, rovina nebo nadrovina (ve vyšších dimenzích) (Müller, Guido 2016).

Jedna z technik, jak z lineární regrese pro normální klasifikaci vytvořit lineární regresi pro víceúrovňovou klasifikaci je metoda zvaná jeden-proti-zbytku. Za využití této metody se binární model učí pro každou třídu, která se pokouší oddělit tuto třídu od všech ostatních tříd, což má za následek tolik binárních modelů, kolik existuje tříd. K provedení predikce jsou všechny binární klasifikátory spuštěny v testovacím bodě. Klasifikátor, který má nejvyšší skóre ve své jediné třídě „vyhrává“, a toto označení třídy je vráceno jako předpověď (Müller, Guido 2016).

5.1.6 **Rozhodovací a regresní stromy**

Metody pro strojové učení založené na rozhodovacích stromech jsou už dlouhou dobu velmi populární. Za tuto popularitu tyto modely vděčí jejich jednoduchosti a síle při předpovídání výsledků (Dean 2014).

Základní metoda při tvoření rozhodovacího stromu je následující. Podíváme se na vstupní proměnné a jejich odpovídající úrovně u dat, nad kterými se má provést analýza, abychom byli schopni poté určit vhodné body rozdělení. Každý bod je poté vyhodnocen, zdali je nejvhodnější pro rozdělení. Nejvhodnější v tomto případě znamená bod, který vytvoří nejčistší podskupinu dat. Proces se poté několikrát opakuje, dokud není splněna podmínka zastavení (Dean 2014).

Vytváření rozhodovacích stromů na základě výše uvedené metody může vést k problému přetrénování. Pokud se snažíme docílit toho, abychom každý bod měli co nejčistší, dojdeme k zbytečně složitým a komplexním modelům rozhodovacích stromů, které jsou až příliš "vhodné" na trénovací data set. Přítomnost čistých listů znamená, že daný model na tréninkové sadě je 100% přesný což je nežádoucí.

Je také možné použít stromy pro regresní úlohy pomocí přesně stejné techniky. Abychom mohli předpovědět, projdeme strom na základě testů v každém uzlu a najdeme list, do kterého spadá nový datový bod. Výstupem pro tento datový bod je průměrný cíl tréninkových bodů v tomto listu (Müller, Guido 2016).

5.1.7 **Hluboké učení (neuronové sítě)**

Hluboké učení, také známe jako hluboké strukturované učení nebo neuronové sítě, je dalším z odvětví strojového učení, které je založeno na algoritmech pokoušejících se najít vzory a modely v komplexních datech. Hluboké učení má mnoho uplatnění od automatického rozpoznání řeči, přes rozpoznávání obrazu až po vyjadřování vztahů v komplexních datech (Shamsuddin 2016). Abychom pochopili, co je neuronová síť, popíšeme si jednotlivě její základní stavební jednotky a metody, které se požívají při učení neuronových sítí:

Neuron: Neuron se většinou skládá ze 4 fundamentálních částí: vstupy, výstupy, váha (anglicky weight) a přenosová funkce (někdy nazývaná také aktivační funkce). Funkce neuronu je velmi prostá: Neuron provede vážený součet všech jeho vstupů, následnou sumu předá přenosové funkci a výsledek se poté předá na výstupy neuronu. Tyto výstupy jsou pak připojené na další neurony (pokud se nejedná o výstupní neuron celé sítě). (Kubat 2017)
Výstup neuronu pak můžeme vyjádřit rovnicí:

$$y = f\left(\sum_{i=1}^{n} w_i * x_i\right)$$

$y \dots výstup\ neuronu$
$w_i \dots váha\ vstupu$
$f \dots přenosová\ funkce$
$x_i \dots vstup\ neuronu$

Neurony můžeme dělit do tří základních kategorií, na základě umístění v neuronové síti: vstupní, skryté a výstupní neurony.

Přenosová funkce: taktéž někdy nazývaná aktivační funkce nebo transformační funkce, převádí na základě matematické rovnice vstupní signál na výstupní signál neuronu. Rovnice může být jakéhokoliv charakteru, mezi čtyři základní přenosové funkce patří: jednotkový skok, Sigmoidova, Gausova, částečně lineární, a lineární přenosová funkce (Sayad 2021).

Vrstva: neuronové sítě se skládají z neuronů, které jsou navzájem propojeny a seřazeny do takzvaných vrstev. Na základě vrstev můžeme také udávat hloubku dané neuronové sítě. Vrstvy můžeme obecně rozdělit do tří typů: vstupní vrstva, kde jsou neuronům předávaný vstupní parametry neuronové sítě, skrytá vrstva neuronů, kde probíhá většina kalkulací, a nakonec výstupní vrstva, kde můžeme vidět výstupní hodnoty neuronové sítě. Jednou z nejpoužívanějších neuronových síti je třívrstvá neuronová síť. V poslední době jsou ale také velmi populární čtyř nebo pětivrstvé neuronové sítě, které našly své uplatnění pro detekci vlastností obrazu a procesování jazyka (Dean 2014).

Základní trénovací metody: Abychom mohli úspěšně trénovat neuronové sítě, potřebujeme metody pro výpočet nejlepší možné váhy, která minimalizuje vysoce nelineární objektivní funkci pro dataset trénovacích dat. Mezi dvě základní metody patří metoda gradientního sestupu a Newtonova metoda. Obě metody vyžadují neustálý vypočet gradientu cílové funkce (Dean 2014).

- **Metoda gradientního sestupu**: Gradient je synonymum pro sklon a sklon ve své normální podobě na XY grafu představuje vzájemný vztah dvou proměnných. Sklon, který v našem případě je pro nás důležitý, popisuje vztah mezi chybou sítě a jednou určitou váhou. Každá váha je pouze jedním faktorem v celé neuronové síti, která zahrnuje mnoho různých transformací, takže můžeme využít řetězové pravidlo,

abychom postupovali zpět z výstupu neuronů do přenosových funkcí až k dané váze, a zjistili její vztah k celkové chybě neuronové sítě. Na základě tohoto zjištění můžeme pak specifickou váhu jemně změnit, aby byla neuronová síť přesnější (Nicholson 2020).

- **Newtonova metoda**: využívá k nalezení minimální hodnoty cílové funkce první derivaci (gradient) a druhou derivaci (zakřivení). V každém svém kroku vypočítá jednoduchou kvadratickou funkci se stejným gradientem a zakřivením jako aktuální cílová funkce a najde nové váhy, které by tuto kvadratickou funkci minimalizovaly. Tento proces se opakuje, dokud se gradient cílové funkce nestane téměř nulovým (Dean 2014).

5.2 Učení bez učitele

Druhá kategorie algoritmů strojového učení je učení bez učitele. Učení bez učitele zahrnuje všechny druhy strojového učení, kde není předem znám žádný výstup a není přítomný žádný supervizor (dohlížející nebo učitel), který by instruoval učící se algoritmus. U tohoto druhu strojového učení se danému algoritmu předají pouze vstupní data a vyžaduje se extrahování znalostí z těchto vstupních dat (Müller, Guido 2016).

Proto se většinou učení bez učitele využívá jako nástroj pro poznání daného datasetu, extrahování znalostí o datech a také k jejich pokročilé vizualizaci (Erl, Khattak, Buhler 2015).

5.2.1 Klastrová analýza

Jak už bylo řečeno, tak na rozdíl od učení s učitelem učení bez učitele dokáže objevit užitečné vlastnosti datasetu. Jedna z fundamentálních metod u učení bez učitele je klastrová analýza, která hledá klastry, nebo skupiny, v daném datasetu. Geometrický střed (centroid) těchto klastrů může být potom využit jako nová vlastnost pro prediktivní algoritmy učení s učitelem nebo například také jako pokročilé vizualizační nástroje (Kubat 2017).

5.2.1.1 ***K-mean algoritmus***

Nejjednodušší algoritmus pro klastrovou analýzu je takzvaný k-mean algoritmus. Písmeno "K" v názvu označuje požadovaný počet klastrů dodaný uživatelem. V prvním kroku algoritmu se vytvoří právě "K" klastrů tak, aby žádný vzorek nezapadal do více než jednoho klastru. Po tomto kroku jsou souřadnice všech geometrických středů (centroidů) vypočítány a je vypočtena vzdálenost jednotlivých vzorků dat od středu centroidů. Nejbližší centroid pak definuje shluk, do kterého by měl vzorek dat patřit. Pokud se už vzorek dat v prvním kroku přiřadil do tohoto klastru, nemusí se dále nic dělat, pokud ne tento vzorek je přiřazen do správného klastru a znovu se vypočítají centroidy. Jakmile algoritmus propočítá každý vzorek dat, jako výstup dostaneme data rozřazená do jednotlivých skupin (Kubat 2017).

5.2.1.2 ***Hierarchická agregace***

Autorka Valentina Alto (2019) rozděluje tento algoritmus na dvě různé techniky: aglomerativní (Agglomerative) a rozporuplné (Divisive). Rozdíl v těchto technikách je směr, kterým algoritmus postupuje – základní koncept je ale naprosto stejný jako u k-mean algoritmu i zde uživatel definuje hodnotu "K" což reprezentuje počet klastrů.

- Aglomerativní technika rozdělí data do "N" klastrů, kdy "N" představuje počet datových vzorků. Následně pak rozděluje klastry, dokud nedojde výsledku "K" klastrů.
- Druhá technika naopak začíná od pouze jednoho klastru a postupně rozděluje tento klastr, dokud nedojde k požadované hodnotě "K" klastrů.

Tento algoritmus se často využívá pro vizualizaci všech možných klastrů a nabízí na ně nový pohled (Müller, Guido 2016).

5.2.2 **Transformace dat a redukce dimenzí**

Sady algoritmů pro transformaci dat bez učitele vytvářejí novou reprezentaci daného datasetu. Tato nová reprezentace dat by měla být pro lidi nebo jiné strojové algoritmy lépe pochopitelná ve srovnání s původními daty.

Běžná aplikace těchto algoritmů je redukce rozměrů vstupního datasetu, kdy vstupní dataset má většinou vysoký počet dimenzí. Transformační algoritmy dokážou tento vysoký počet dimenzí zredukovat a najdou novou a jednodušší reprezentací dat, která shrnuje základní charakteristiky původního datasetu. Jedno z využití těchto metod je snížení dimenzí datasetu na pouhé dvě dimenze za účelem vykreslení dat do grafu (Müller, Guido 2016).

Za počet dimenzí v datasetu považujeme počet vlastností (features) datasetu, tyto dimenze vlastností jsou pak reprezentovány sloupci v datasetu. Ve většině případů jsou tyto sloupce korelované, a v tom případě jsou tyto informace nadbytečné a můžou negativně ovlivnit účinnost a efektivnost algoritmů strojového učení. Proto má toto odvětví učení bez učitele v datové vědě své důležité místo (Roman 2019).

5.2.2.1 ***Algoritmy pro výběr vlastností***

Algoritmy pro výběr vlastností datasetu umožní prediktivním modelům strojového učení být mnohem přesnější. Tyto algoritmy umožní výběr takových vlastností datasetu, které poskytnou stejně dobrou nebo ještě lepší přesnost při míň datech. Dále lze využít tyto metody k identifikaci a odstranění nepotřebných, irelevantních a nadbytečných atributů z dat, která prediktivnímu modelu nepřispávají k jeho přesnosti nebo mu dokonce přesnost snižují. (Brownlee 2014).

Autor Brownlee Jason (2014) dále tyto algoritmy rozděluje na tři základní metody: filtrovací (filter), balení (wrapper) a integrované metody (embedded). Dále autor tyto metody také popisuje:

- **Filtrovací metody**: tato metoda využívá statistická měřítka k přiřazení takzvaného "hodnocení" ke každé vlastnosti datasetu. Tyto vlastnosti jsou poté seřazeny podle tohoto skóre a jsou buď vybrány, nebo odstraněny z datasetu.
- **Balící metody**: vnímají výběr vlastností jako problém vyhledávání, kde jsou připravené kombinace vlastností vyhodnocovány a porovnávány s jinými kombinacemi. Následně se využije prediktivní model k vyhodnocení dané kombinace vlastností na základě přesnosti daného modelu. Výběr nejlepší kombinace a nejlepšího modelu může být metodický a nemusí to vybrat sám uživatel.
- **Integrované metody**: nebo také nazývány taktéž vestavěné metody, se učí, která vlastnost datasetu nejlépe přispívá k přesnosti modelu během jeho vytváření. Nejběžnějším typem integrovaných metod jsou regularizační metody (nebo také nazývané penalizační metody).

5.2.2.2 Algoritmy pro extrakce vlastností

Na webových stránkách Elite Data Science (2019) se dočteme, že algoritmy pro extrakci vlastností slouží k vytvoření nové, menší sady vlastností, která stále zachycuje většinu užitečných informací. Algoritmy pro výběr vlastností vyberou podmnožinu originálních vlastností, naopak algoritmy pro extrakci vlastností vytváří vlastnosti naprosto nové. Autor Roman Victor (2019) uvádí tři základní algoritmy pro extrakci vlastností, analýza hlavních komponent (PCA), náhodná projekce (random projection) a analýza nezávislých komponent (ICA):

Analýza hlavních komponent (PCA): tato metoda rotuje datovou sadu takovým způsobem, že rotované prvky jsou statisticky nekorelované. Po tomto kroku často následuje výběr podmnožiny vlastností na základě jejich důležitosti pro daný dataset (Müller, Guido 2016). PCA v podstatě kombinuje všechny funkce rotací a automaticky extrahuje ty nejdůležitější. Jedná se o systemizovaný způsob transformace datových vlastností na jejich hlavní součásti, které jsou pak možné využít jako zcela nové vlastnosti (Roman 2019).

Náhodná projekce: je ještě výkonnější a efektivnější metoda pro redukci dimenzí než PCA. Je zcela normálně využívána u datasetů s nadměrným množstvím vlastností. Náhodná projekce sníží počet dimenzí datasetu jeho přenásobením na náhodnou matici. Tímto se dataset promítne do nového podprostoru jeho vlastností (Roman 2019).

Analýza nezávislých komponent (ICA): stejně jako u předešlých metod tak i u ICA metody se snažíme přijít na nové užitečné vlastnosti datasetu. Na rozdíl od náhodné projekce a PCA se ICA nesnaží maximalizovat rozptyl nových funkcí. ICA přepokládá, že vlastnosti datasetu jsou kombinace nezávislých zdrojů a tyto zdroje se snaží izolovat jako nové vlastnosti. ICA tedy identifikuje mezi původními vlastnosti ty které nezávisle přispívají k datové sadě (ty které mají minimální korelaci s ostatními vlastnostmi) (Roman 2019).

6 Zpracování konkrétního datasetu

6.1 Výběr nástrojů pro zpracování datasetu

Jeden z nejpopulárnějších jazyků, který se používá pro analýzu velkých dat strojovým učením, je skriptovací jazyk Python. Zároveň je celý ekosystém knihoven Python volně dostupný a je tedy možné tyto knihovny volně využívat a čerpat tak ze zkušeností rozsáhlé komunity programátorů. Z těchto důvodů je pro účel této práce zvolen jazyk Python. Tento programovací jazyk a populární knihovny jsou popsané v kapitole *3.2.4 Python*.

Python nativně má velmi pomalé a slabé nástroje pro manipulaci a načítání větších dat. Proto byla volena knihovna *Pandas* pro pracování s daty v nejrůznějších formátech a také pro jednoduchou manipulaci s těmito daty skrz velmi intuitivní rozhraní objektu *DataFrame*. Vizualizace v praktické částí této bakalářské práce byla docílena pomocí populární knihovny *Matplotlib*, která umožňuje generování obrázků v publikační kvalitě. Pro pokročilejší obrazce, kde byla nutnost zpracování datových struktur typu graf, byla využita grafická knihovna *networkx*. Knihovna *networkx* byla konkrétně využita k vygenerování obrázků č.8 a č.9, které popisují interní schéma zkoumané produkční linky. Zpracovávaný dataset v praktické části je nadstandartní velikosti a jelikož programovací jazyk Python nativně pracuje pouze s jedním jádrem procesoru zařízení, je nutné zapojit knihovny, které umožní využívat plný potenciál procesoru koncového zařízení. Pro tyto účely byly vybrány knihovny *Dask* a *Joblib*, které umožňují využívání všech jader procesoru a paralelní běh procesů pro načítání dat ale také pro učení algoritmů strojového učení. Hlavní knihovna pro algoritmy a prediktivní modely strojového učení byla zvolena *sklearn*. Z této knihovny můžeme využívat algoritmy pro selekci proměnných, manipulaci s datasetem a tak také pro tvorbu prediktivních modelů. Další knihovny, které byly využity pro tuto bakalářskou práci, jsou: *numpy*, *timeit*, *json* a *collections*.

Programové prostředí pro praktickou část bakalářské práce bylo zvoleno *JupyterNotebook* s distribucí *Anaconda*. Přiložené zdrojové kódy tedy mají v sobě obsažené také veškeré generované obrazce a výstupy jednotlivých funkcí což zaručuje jednoduchou přehlednost a snadné prohlížení.

6.2 Výběr datasetu

Většina datasetů nese podobu souborů ve formátu csv, které jsou popsány v kapitole *4.1.3 Formát dat*. Toto však není pravidlem a dataset může existovat v jakémkoli formátu dat a může se skládat z více souborů. Tyto soubory se dají generovat nebo získat různými procesy, jako je například tvorba datasetu sběrem dat z inteligentních čidel, metodou zvanou *web scraping* (tzn. sběr dat z webu) nebo jednoduše využitím webových stránek jako je Twitter, Youtube nebo Facebook, které svá data mají veřejně přístupná.

Tato práce se nezabývá tvorbou datasetu, ale pouze analýzou dat. Proto je klíčové podívat se na místa, kde daný lze dataset najít a následně si vybrat ten, který zapadá do kontextu Průmyslu 4.0 a je vhodný pro analýzu strojovým učením.

6.2.1 **Zdroje velkých dat**

Na internetu nalezneme mnoho stránek poskytujících veřejné datasety ke stažení. Paruchi Vik (2020) ve svém článku popisuje několik zdrojů datasetů rozdělených do kategorií na základě využití dat, které poskytují.

Stránky jako *fivethirtyeight.com*, *buzzfeed.com* nebo *nasa.com* poskytují vhodné datasety pro projekty zaměřené na vizualizaci dat. Datasety na těchto stránkách většinou bývají velmi přesné a bez nevhodného šumu, který by měl negativní vliv na vizualizaci. Jednotlivé vlastnosti datasetu jsou ve většině případů velmi dobře popsané, takže vizualizace může být velmi přesná.

Pro datasety určené k analýze strojovým učením je žádoucí, aby existoval určitý vztah mezi jednotlivými vlastnostmi datasetu. Tato podmínka zde vyvstává hlavě kvůli tomu, že chceme u těchto datasetů vytvářet prediktivní modely, které to vyžadují. Následující stránky nám takovéto datasety mohou poskytnout: *kaggle.com*, *quandl.com* nebo na stránkách Machine Learning Repository pod odkazem *archive.ics.uci.edu*.

6.2.2 **Kaggle: Výběr konkrétního datasetu**

Pro účely této práce je vybrán dataset ze stránek *Kaggle* dostupný z url: *www.kaggle.com/c/bosch-production-line-performance/data*. Stránky nabízejí široký výběr datasetů s nejrůznějšími typy dat. Za stránkami stojí také nemalá komunita datových vědců, kteří sdílejí své zkušenosti a metody zpracování konkrétních datasetů. Některé datasety jsou zde uvedeny v rámci soutěže o nejlepší analýzu vybraného datasetu. Proto jsou pro účely této práce k dispozici i poznatky z diskuse soutěžících. Zveřejňují zde své metody, zdrojové kódy, píší o vlastnostech datasetů, vhodných metodách a zmiňují další užitečné informace, které jsou veřejně přístupné a lze tedy z nich čerpat a použít je pro možné srovnání závěrů této práce.

Jak již bylo zmíněno, stránky *Kaggle* disponují velkou knihovnou nejrůznějších datasetů. Z oblasti mnou studovaného oboru Počítačové systémy pro průmysl 21. století neboli Průmysl 4.0 bylo nalezeno 79 možných datasetů. Pro účely této bakalářské práce je vybrán dataset generovaný produkční linkou firmy Bosch.

Ze 79 datasetů byly pouze 3 soutěžní a lze tak u nich dohledat rozsáhlejší diskuse účastníků o způsobech analýzy dat. Největší soutěž se týkala detekce defektů kovových materiálů. Dataset byl však velmi rozsáhlý a vyžadoval určité znalosti dané tématiky, což je nad rámec této práce.

Dataset, který je pro práci vybrán, je dataset firmy Bosch (2016). Tento dataset je jeden z největších datasetů z pohledu počtu jeho vlastností, který můžeme na platformě Kaggle najít.

To z něj dělá jeden z nejnáročnějších datasetů pro analýzu. Díky své komplexnosti je vhodný pro implementaci vybraných algoritmů a metod.

6.2.3 **Stanovení cílů**

Firma Bosch (2016) hodnotí jednotlivé prediktivní modely na základě MCC (Matthewsova korelačního koeficientu). Rovnice pro výpočet MCC má následující tvar:

$$MCC = \frac{(PP - PN) - (FP * FN)}{\sqrt{(PP + FP)(PP + FN)(PN + FP)(PN + FN)}}$$

MCC je často využívaná metoda ve strojovém učení pro vyhodnocení binárních klasifikačních modelů, tedy modelů, kde se výsledek určuje pouze jako pozitivní nebo negativní (1 nebo 0). MMC je hodnota, která se pohybuje mezi -1 a +1. Pokud MCC je roven +1 jedná se o perfektní klasifikaci, kdy prediktivní model dokázal předpovědět všechny případy správně, naopak hodnota -1 poukazuje na kompletní selhání prediktivního modelu, kdy model nepředpověděl správně ani jednu hodnotu (Chicco, Jurman 2020).

PP hodnota reprezentuje počet pravdivě pozitivních výsledků. Za pravdivě pozitivní výsledek považujeme situaci, kdy predikovaná hodnota je pozitivní a taktéž hodnota skutečná je pozitivní. Stejně jako u PP tak PN (pozitivně negativní) výsledky jsou hodnoty, kdy se predikovaná hodnota shoduje se skutečnou hodnotou, jediný rozdíl je, že v tomto případě se nejedná o pozitivní hodnotu, ale negativní. FP znamená falešně pozitivní výsledek, což je neshoda predikce a skutečné hodnoty, kdy skutečná hodnota je pozitivní. Naopak u falešně negativního výsledku, FN, se jedná o neshodu predikce se skutečnou hodnotou, kdy skutečná hodnota je negativní (Chicco, Jurman 2020).

6.2.4 **Dokumentace firmy Bosch**

Firma Bosch (2016) popisuje, že dataset se skládá z obrovského množství anonymních vlastností. Což pro tuto práci znamená, že dopředu u vlastností datasetu neznáme jejich skutečnou funkci na výrobní lince. Jednotlivé vlastnosti jsou pojmenovány podle jejich umístění ve výrobním cyklu. Například značení vlastnosti *L1_S2_F3* znamená, že se daný prvek nachází na první výrobní lince, druhé servisní stanici a z celkových prvků na této stanici je třetí v pořadí. Toto značení vychází z anglických názvu Line (výrobní linka), Station (stanice) a Feature (vlastnost).

Dále je dataset rozdělený celkem do šesti souborů na základě typu vlastností, které obsahují: numerické (numeric), kategorické (categorical) a nakonec časová data (date feature). Soubory s časovými daty říkají, kdy byly dané datové vzorky naměřeny a uloženy do datasetu. Firma Bosch už také datasety rozdělila na trénovací data a testovací data. Trénovací data jsou ta, na kterých se mají prediktivní modely učit. Testovací data slouží pro ověření přesnosti daných modelů.

6.3 Numerické data

První typ dat v datasetu jsou numerická data. Tato část datasetu je nejmenší, a proto začneme s prozkoumáváním dat u této kategorie. Abychom se mohli podívat na dimenze datasetu a jednotlivé vlastnosti, je zapotřebí načíst pouze část dat do paměti RAM. Zde si můžeme všimnout hned tří klíčových informací:

- V datasetu se nacházejí nedefinované hodnoty. Knihovna Pandas, kterou nyní využíváme pro prozkoumávání dat, tyto hodnoty reprezentuje jako NaN hodnotu. Je nutné uvést, že tato hodnota se nerovná sama sobě a také, že není rovná nule.
- Numerický dataset má vlastnost Response, toto je naše cílová proměnná, kterou se budeme snažit predikovat a kolem které budeme trénovat a vytvářet naše prediktivní modely.
- Počet vlastností v tomto datasetu je 970 (sloupce). Můžeme zde najít vlastnost Id, díky které můžeme identifikovat jeden výrobek napříč všemi soubory v datasetu. Dále je zde již zmiňovaná Response vlastnost. Zbylých 968 sloupců jsou data z čidel z výrobní linky, která jsou anonymní, a nevíme, co reprezentují. Neznáme ani fyzikální rozměr jejich hodnot.

	Id	L0_S0_F0	L0_S0_F2	L0_S0_F4	L3_S51_F4258	L3_S51_F4260	L3_S51_F4262	Response
0	4	0.030	-0.034	-0.197	NaN	NaN	NaN	0
1	6	NaN	NaN	NaN	NaN	NaN	NaN	0
2	7	0.088	0.086	0.003	NaN	NaN	NaN	0
3	9	-0.036	-0.064	0.294	NaN	NaN	NaN	0
4	11	-0.055	-0.086	0.294	NaN	NaN	NaN	0
...	...	...	...	...	...	...	...	...
95	192	0.069	0.131	0.330	NaN	NaN	NaN	0
96	197	NaN	NaN	NaN	NaN	NaN	NaN	0
97	200	-0.036	-0.026	-0.161	NaN	NaN	NaN	0
98	201	0.030	0.056	0.312	NaN	NaN	NaN	0
99	205	-0.108	-0.086	-0.179	NaN	NaN	NaN	0

Tabulka 6.1: *tabulka vlastností numerických dat (redukovaná)*

Tabulka 6.1 demonstruje, že se u numerických dat nacházejí NaN hodnoty. Firma Bosch neuvedla, co tyto hodnoty znamenají, dedukcí však můžeme dojít ke dvěma logickým vysvětlením těchto hodnot:

- Pokud nalezneme hodnotu NaN v řádku, který reprezentuje výrobní cyklus výrobku, lze říct, že se jedná o neúplná data, která jsou tudíž nevalidní pro účely této analýzy.
- Hodnota NaN nám o výrobku říká, že výrobek neprošel daným čidlem při svém výrobním cyklu. Data jsou tedy validní a můžeme s nimi pracovat.

Pro správné chápání datasetu musíme dojít k jednomu ze dvou uvedených závěrů. Docílíme toho pomocí sestrojení jednoduchého kódu, který eliminuje veškeré řádky s hodnotami NaN. Výsledkem tohoto kódu je však prázdný dataset. To znamená, že každý řádek obsahoval minimálně jednu instanci hodnoty NaN a proto byl odstraněn.

Toto nás vede k jednoznačné dedukci, že řádky s hodnotou NaN jsou validní a měly by být zachovány veškeré řádky, kde se tyto hodnoty nacházejí. Přesto se v datasetu nacházejí řádky, které mají veškeré hodnoty NaN. Zde nelze získat nějakou informaci a je tedy zcela relevantní tyto data odstranit a nepracovat s nimi.

Firma Bosch uvádí konvenci, podle které jsou jednotlivé vlastnosti pojmenovávány (tato konvence je podrobněji popsána v kapitole *6.3.3 Porozumění datasetu*). Lze tedy zjistit, kolik jednotlivých linek, stanic a vlastností se v této kategorii nachází. Tuto informaci je možné z datasetu získat kódem, který bude iterovat skrze všechny sloupce datasetu. Z každého názvu sloupce si zapamatuje unikátní název linek, stanic a vlastností. Výstup tohoto kódu nám prozradí, že se v numerických datech nacházejí 4 linky, 50 stanic a 968 různých čidel (nebo také vlastností). Dále je možné zjistit informace o rozdělení stanic mezi jednotlivé výrobní linky:

```
Informace o jednotlivých linkách:
Linka L0 má 24 stanic   [S0 až S23]
Linka L1 má 2 stanic    [S24 až S25]
Linka L2 má 3 stanic    [S26 až S28]
Linka L3 má 21 stanic   [S29 až S51]
```

Obrázek 6.1:*rozdělení stanic mezi výrobní linky*

Na obrázku 6.1 je možné vidět, že největší linky jsou L0 a L3, kde se nachází 90 % veškerých stanic. Další zajímavá informace z těchto dat je pojmenovávání stanic. I když je v této kategorií pouze 50 stanic, nacházíme zde stanici s názvem S51. Můžeme předpokládat, že stanice, které nebyly nalezeny v této kategorii, jsou uloženy v kategorických datech. Následně můžeme získat data o počtu čidel v jednotlivých stanicích, zde zjistíme, že stanice S24 a S25 jsou největší stanice s 229 a 284 čidly.

6.3.1 **Redukce velikosti dat**

Jelikož je při vizualizaci dat vhodné počítat se všemi daty najednou, je žádané, abychom celý dataset nahráli najednou do paměti. Problém je ale v tom, že velikost vybraného datasetu nám neumožňuje nahrát veškerá data najednou do RAM paměti. Naskytne se nám několik metod, které lze vyzkoušet a kombinovat pro co nejrychlejší a nejúspornější práci s daty:

- Kouskování (chunking): zpracování datasetu v menších a zvládnutelných blocích.
- Změna datových typů: knihovna Pandas nepřiřazuje načteným datům paměťově nejúspornější datový typ. Můžeme jej tedy manuálně změnit.
- Paralelismus: načtení dat paralelně s kompletním využitím procesoru pomocí knihovny Dask (tato metoda však řeší jenom čtecí rychlost, nikoliv redukci velikosti).

Kouskování datasetu je možno aplikovat na jakýkoliv dataset, tato metoda je však složitější na implementaci. Některé funkce trvají i desítky minut, což není efektivní řešení. Než tedy data zredukujeme a využijeme paralelní čtení, musíme dataset nejprve celý zpracovat touto metodou.

Kouskovací metodou zpracujeme celý dataset a zjistíme maximální a minimální hodnoty pro každý sloupec. Přicházíme na klíčové informace – data v tomto datasetu jsou normalizovaná. Můžeme zde najít pouze hodnoty v intervalu ⟨-1, +1⟩. Většina hodnot má 3 desetinná místa. Ojediněle vidíme hodnotu, která má až 14 desetinných míst. Tyto hodnoty jsou většinou periodické, kdy jejich poslední číslice má jinou hodnotu:

```
'L1_S24_F829': [-0.373, 0.47],
'L1_S24_F834': [-0.898, 0.035],
'L1_S24_F839': [-0.40399999999999997, 0.32899999999999996],
'L1_S24_F844': [-0.47, 0.519],
'L1_S24_F857': [-0.5670000000000001, 0.41600000000000004],
'L1_S24_F862': [-0.5429999999999999, 0.43200000000000005],
'L1_S24_F867': [-0.93, 0.07],
'L1_S24_F872': [-0.39899999999999997, 0.601],
```

Obrázek 6.2:*ukázka intervalu jednotlivých čidel*

Informace, které se pohybují řádově za 3 desetinným místem, jsou pro nás zcela irelevantní a nesou minimální informativní hodnotu. Proto je můžeme odstranit a ušetřit tím místo v paměti.

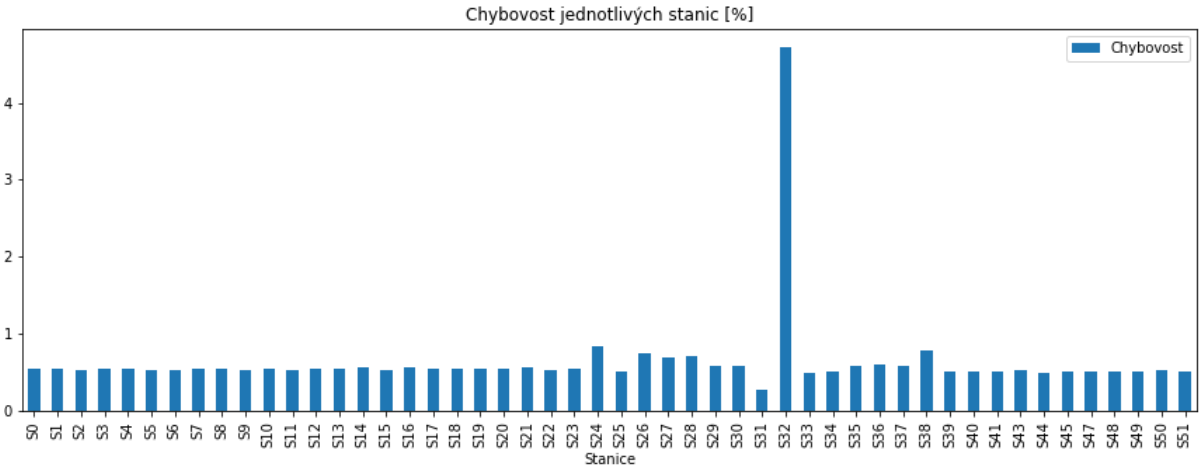
Knihovna Pandas nám poskytuje možnost předdefinovat datové typy pro jednotlivé sloupce. Pro sloupce čidel (tedy všechny sloupce kromě Id a Response) předefinujeme datový typ na float16. Tento datový typ může reprezentovat hodnoty v intervalu ⟨-65504, +65504⟩s rozlišením až +0,0010004 a s přesností 3 desetinných míst. Tento datový typ je tedy zcela ideální pro naše data. Když aplikujeme zmíněnou metodu, tedy změnu datových typů, tak se nám numerická data načtou do paměti RAM a zabírají méně než 3GB paměti. Když tuto metodu zkombinujeme spolu s paralelismem implementovaným prostřednictvím knihovny Dask, docílíme načtení celých numerických dat do paměti RAM v řádu desítek sekund.

6.3.2 **Vizualizace**

První věc, kterou lze jednoduše vizualizovat v tomto datasetu, je chybovost jednotlivých čidel (tedy pravděpodobnost selhání produktu, jestli daným čidlem prošel). Jelikož pracujeme s 968 vlastnostmi datastu, vizualizace je poněkud nepřehledná, až nečitelná. Z tohoto důvodu

vizualizujeme pouze chybovost jednotlivých stanic. V této kategorii je 50 stanic, což je ještě počet, který lze do grafu jednoduše vyobrazit.

Kdybychom data zpracovávali bez metod a redukcí dat popsaných v předešlé podkapitole, trvala by nám celá kalkulace v průměru jednu hodinu. Když ale tyto metody aplikujeme, můžeme načíst celý dataset a také ho zpracovat za méně než 3 minuty. Výsledkem je graf chybovostí jednotlivých stanic ve výrobním procesu.

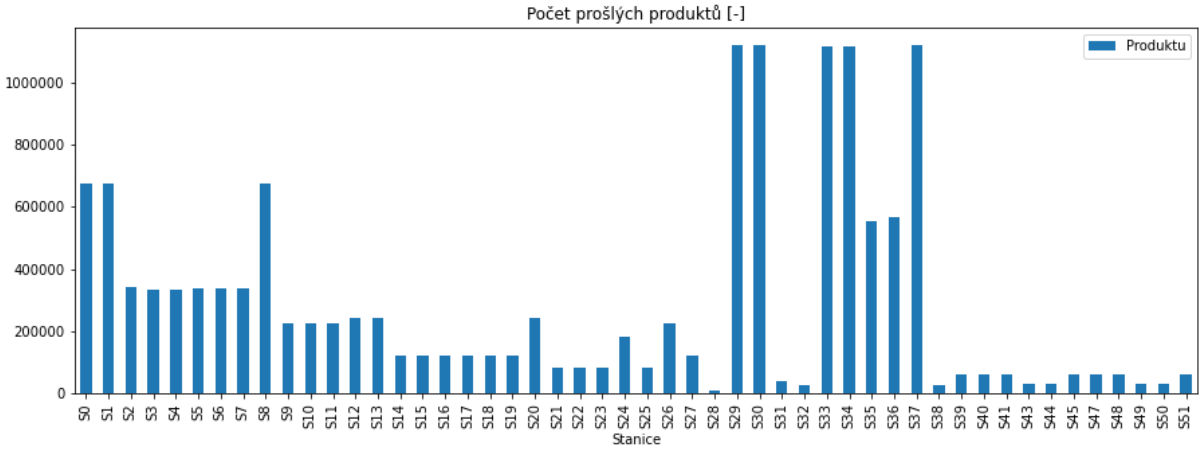


Obrázek 6.3: *chybovost jednotlivých stanic*

Z předešlého obrázku (obr. 6.3) je zřejmé, že stanice S32 má mnohonásobně větší chybovost než ostatní stanice. Vzhledem k tomu, že jsou zkoumané vlastnosti anonymní, nevíme jakou má stanice funkci. Můžeme ale odhadovat, což potvrzují i diskuse u datasetu na stránkách Kaggle, že se jedná o opravnu na výrobní lince.

Druhým extrémem je stanice S31. Zde můžeme najít nejmenší chybovost z celého výrobního procesu. Tyto informace jsou vhodným podkladem pro fázi tvoření nových vlastností a výběr vlastností. Lze se domnívat, že stanice S32 a S31 uchovávají v sobě silnou informaci o tom, zda daný výrobek bude vadný nebo projde kontrolou.

Další graf k vykreslení je počet produktů, které prošly skrz danou stanici:

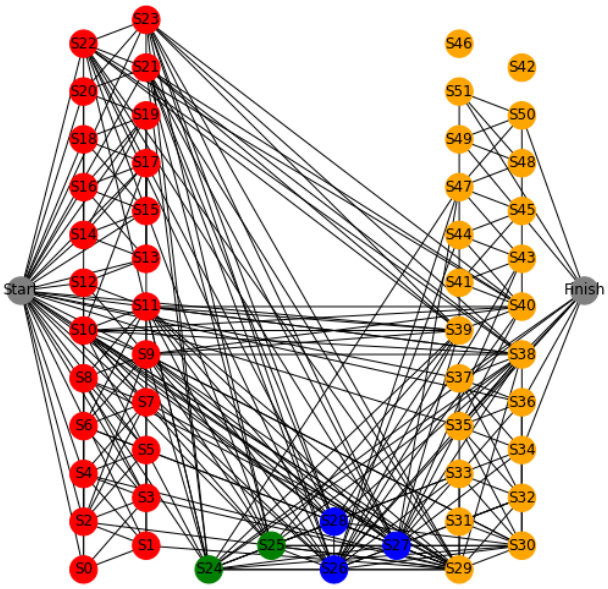


Obrázek 6.4:*počet produktů na stanici*

Určitá diverzita mezi jednotlivými výrobky stojí za povšimnutí. Z obrázku (obr. 6.4) je zřejmé, že ne všechny výrobky začínají svůj výrobní cyklus na stanici S0. Zároveň zde vidíme, že všechny produkty projdou stanicemi S29, S30, S33 S34 a S37.

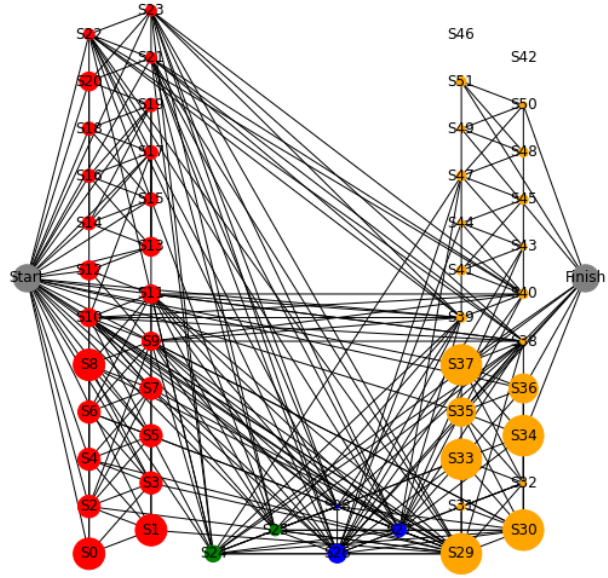
Stanice S31 a S32 mají jeden z nejmenších počtů prošlých produktů. Zato ale víme z předešlého grafu, že mají největší a nejmenší chybovost. Toto nám opět napovídá, že tyto stanice nesou silnou informaci o výrobku, což můžeme zužitkovat při selekci proměnných v dalších krocích.

Další vizualizace vykresluje extrakci různých výrobních cyklů. Z těchto dat můžeme zjistit, zdali výrobní linky vyrábí malý počet produktů a můžeme tuto informaci použít pro tvoření nových vlastností nebo je počet druhů produktů příliš velký pro tvoření nových vlastností. První obrázek (obr. 6.5) demonstruje pouze výrobní cesty produktů, tedy bez dalších informací navíc. Pro lepší přehlednost jsou stanice barevně a pozičně oddělené do jednotlivých linek:



Obrázek 6.5:*vizualizace struktury výrobních linek*

V druhém grafu (obr 6.6) je zakomponovaná informace o počtu produktů, které prošly skrz jednotlivé stanice v minulých krocích:



Obrázek 6.6:*vizualizace struktury výrobních linek s počtem produktů na stanici*

6.3.3 **Výběr vlastností**

Dimenze numerických dat je příliš velká pro úspěšné učení prediktivních modelů, proto musíme vybrat vlastnosti s největší informativní hodnotou a ostatní vlastnosti zahodit. Tento proces bude mít velký vliv na výsledky prediktivních modelů, a proto se musí každý krok této analýzy pečlivě promyslet.

6.3.3.1 ***Algoritmy pro výběr vlastností***

Pro tento úkol je zvolena knihovna *sklearn.feature_selection*, která nabízí řadu algoritmů pro výběr informativně nejhodnotnějších vlastností.

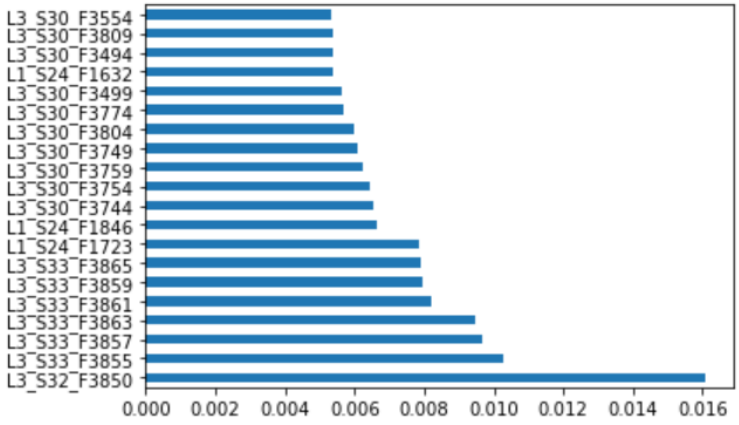
Začneme algoritmem *VarianceThreshold*, který vyřadí vlastnosti datasetu s určitou odchylkou na základě definované prahové hodnoty. Při nastavení prahové hodnoty na 0.5 nenajde algoritmus žádné vlastnosti. Toto indikuje, že hodnoty v numerických datech nejsou moc rozptýlené. Algoritmus při nastavení prahové hodnoty 0.0 taktéž nenajde ani jednu vlastnost. Máme tedy garantované, že veškeré vlastnosti mají určitou informativní hodnotu. Nakonec nastavíme prahovou hodnotu na 0.1, což odpovídá rozkmitu 5 % v numerických datech, algoritmus najde 12 vlastností s větším rozkmitem.

Jako další algoritmy pro výběr vlastností využijeme *SelectKBest* a *SelectPercentil*. Vstupní funkci pro tyto algoritmy použijeme funkce *f_classif*, která se využívá pro klasifikační úlohy. Tato funkce na svůj vstup, akceptuje pouze data bez *NaN* hodnot, což je v našem případě problém. Nahradíme tedy hodnoty *NaN* číslem 0. Toto není úplně ideální řešení, jelikož se v datasetu nacházejí běžně data s nulovou hodnotou. Riskujeme tedy ztrátu informace nebo dokonce zkreslení informace, i když pouze minimální. Vlastnosti, které nám algoritmy vyhodnotily jako nejvíc přínosné, si uložíme do souboru, ať k nim máme později přístup.

6.3.3.2 ***Využití prediktivních modelů***

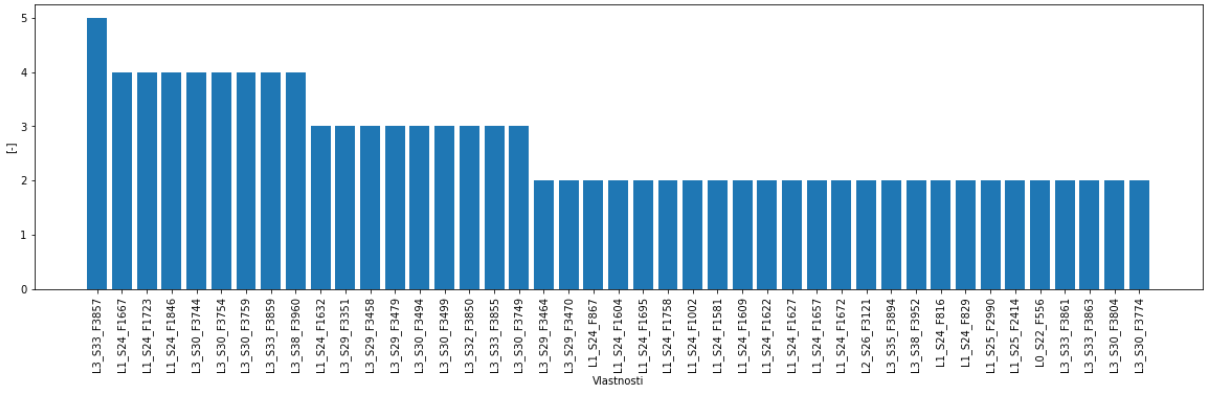
Jedna z metod, jak vybrat silné vlastnosti datasetu, je aplikovat prediktivní model na tento dataset a následně nechat tento model vyhodnotit, které vlastnosti mají největší dopad na predikci. K tomuto úkolu můžeme využít jakýkoliv model strojového učení s učitelem. Pro účely této práce je využito *PassiveAggressiveClassifier*, *LogisticRegression*, *DecisionTreeClassifier*, *RandomForestClassifier* a *ExtraTreesClassifier*.

Numerický dataset je ale stále příliš velký, aby se na něm prediktivní modely dokázaly učit v relativně krátkém čase. Proto jsou vytvořeny bloky datasetu, na kterém jednotlivé modely trénujeme. Nejlepší MCC skóre 0,316 má model *RandomForestClassifier*, který vyhodnotil vlastnosti zobrazené v grafu níže jako nejvíce informativně přínosné. Na obrázku (obr. 6.7) také vidíme, že nejsilnější proměnná se vyskytuje ve stanici S32. Tento výsledek nám opět potvrzuje správnou predikci ohledně fungování vlastností z předešlých kapitol:



Obrázek 6.7:*důležité vlastnosti podle RandomForestClassifier*

Díky algoritmům z knihovny *sklearn.feature_selection* jsme získali seznam nejvýznamnějších a nejvíce informativně přínosných vlastností v numerickém datasetu. Pro vytvoření redukovaného datasetu jsou využity pouze vlastnosti, které se objevily v algoritmech více než dvakrát. Graf níže (obr. 6.8) popisuje výskyt jednotlivých vlastností napříč selektivními algoritmy, a tedy také vlastnosti, které jsou využity pro redukovaný dataset.



Obrázek 6.8:*nejvýznamnější vlastnosti na základě selektivních algoritmů*

6.3.3.3 ***Redukce ztráty informace***

Vždy při redukci dimenze datasetu existuje riziko, že se ztratí nějaká důležitá informace. Abychom tedy do nového datasetu zahrnuli co nejvíce informací z původního souboru, vytvoříme nové proměnné, které budou čerpat svou informaci z celého původního numerického datasetu. Aplikovanou metodou nelze zaručit, že informační hodnota nebude ztracena. Ztrátu můžeme pouze redukovat.

V novém datasetu jsme vytvořili řadu nových vlastností, jako jsou například extrémní hodnoty ve výrobním cyklu výrobku, tedy minimální a maximální hodnota vlastností. Dále jsme vypočítali průměrnou hodnotu vlastností datasetu.

Na základě vybraných vlastností všemi modely strojového učení s učitelem a také všech vlastností vybraných algoritmy z knihovny *sklearn.feature_selection* jsme vytvořili zredukovaný numerický dataset s minimální ztrátou informační hodnoty. Nový redukovaný dataset jsme uložili do speciálního binárního souboru *feather*, pro řádově rychlejší čtecí a zapisovací operace za účelem usnadnění práce s tímto datasetem prediktivním modelům. Z tohoto souboru si můžeme poté vybírat jednotlivé vlastnosti a při trénování prediktivních modelů nalézt nejlepší možnou kombinaci.

6.4 Kategorická data

Kategorická data jsou druhý typ dat v našem datasetu. Velikost této části dat je ještě větší než u numerického typu, a proto si znovu načteme pouze část dat pro získání informací o dimenzích tohoto datasetu. Z tohoto segmentu dat (tabulka 6.2) lze získat tři důležité informace:

- Vyskytuje se zde mnohem více *NaN* hodnot než u numerických dat. Jedná se ale o dedukci z malé části dat, proto je zapotřebí tuto hypotézu dále ověřit.
- Nenachází se zde Response sloupec, ale pouze sloupec Id produktů.
- Počet vlastností je 2141, což je dvakrát víc než u numerických dat.

	Id	L0_S1_F25	L0_S1_F27	L0_S1_F29	L3_S49_F4235	L3_S49_F4237	L3_S49_F4239	L3_S49_F4240
0	4	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	6	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	7	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	9	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	11	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...
95	192	NaN	NaN	NaN	NaN	NaN	NaN	NaN
96	197	NaN	NaN	NaN	NaN	NaN	NaN	NaN
97	200	NaN	NaN	NaN	NaN	NaN	NaN	NaN
98	201	NaN	NaN	NaN	NaN	NaN	NaN	NaN
99	205	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Tabulka 6.2: *tabulka vlastností numerických dat (redukovaná)*

Extrahujeme počet linek, stanic a vlastností stejně jako u numerických dat. Tato data můžeme poté porovnat s numerickými a zjistit, zda narazíme na nové linky, stanice nebo vlastnosti. Stejně jako u numerických dat i zde nalezneme 4 linky. Počet stanic a čidel se ale liší. Máme zde 34 stanic a 2140 čidel. Když uděláme průnik linek, stanic a čidel jak z numerických tak z kategorických dat, dojdeme výsledku, že finální počet linek je 4, počet stanic 52 a celkově se

v datasetu nachází různých 3108 čidel. Rozložení stanic v jednotlivých linkách pro kategorická data je vyobrazeno na obrázku níže:

```
Informace o jednotlivých linkách:
Linka L0 má 15 stanic   [S1 až S23]
Linka L1 má 2 stanic    [S24 až S25]
Linka L2 má 3 stanic    [S26 až S28]
Linka L3 má 14 stanic   [S29 až S49]
```

Obrázek 6.9:*rozdělení stanic mezi výrobní linky (kategorická data)*

Stejně jako u numerických dat i zde můžeme vidět největší počet stanic na linkách L0 a L3. Další analýza kategorických dat nám řekne, že stanice S24, S25 a nově také S30 mají počet čidel nad 100.

6.4.1 **Transformace dat**

Prediktivní modely strojového učení špatně pracují s kategorickými daty, proto existují metody pro převedení kategorických dat na data numerická. Tyto metody také většinou redukují paměťovou náročnost datasetu, jelikož převedou textové hodnoty do numerických hodnot, které zabírají méně místa v paměti.

Před transformováním datasetu musíme ještě získat počet kategorií v kategorických datech a zjistit, jak jsou tyto kategorie reprezentovány. Na obrázku níže (obr. 6.10) lze vidět, že kategorií je v datasetu 93. Každá kategorická hodnota začíná písmenem T. Toto vede mnohé datové vědce na stránkách Kaggle k domněnce, že se jedná o testy prováděné na výrobcích během výrobního cyklu.

```
Počet kategorických hodnot: 93
Všechny kategorie: {'T786432', 'T64', 'T1152', 'T96', 'T-18748192', 'T18436',
'T91764', 'T262144', 'T8651776', 'T97', 'T4', 'T514', 'T2516', 'T-214748294
4', 'T748928', 'T143', 'T33554432', 'T32896', 'T8768', 'T128', 'T-214748268
8', 'T16777216', 'T98', 'T492', 'T331648', 'T83888', 'T12582912', 'T-21474816
64', 'T1372', 'T9', 'T24', 'T16777232', 'T1310', 'T9174', 'T-2147482176', 'T3
2', 'T16512', 'T618624', 'T52', 'T6', 'T5', 'T48576', 'T4718592', 'T524288',
'T56', 'T11141888', 'T268435456', 'T96112', 'T7', 'T16777472', 'T16777557',
'T16', 'T6553', 'T8389632', 'T8912896', 'T-2147483646', 'T16777248', 'T335544
48', 'T48', 'T917', 'T-2147482816', 'T55424', 'T256', 'T678864', 'T25165824',
'T16384', 'T589824', 'T26808', 'T41944', 'T-21474872', 'T134217728', 'T17825
8', 'T16779428', 'T145', 'T36992', 'T113776', 'T2', 'T-2147482432', 'T8', 'T3
3554944', 'T-21474819', 'T-2147483647', 'T65536', 'T1132', 'T1', 'T86752', 'T
786944', 'T-21474825', 'T524544', 'T63616', 'T3', 'T512', 'T-2147483648'}
```

Obrázek 6.10: *všechny kategorie*

Čísla v kategoriích nabývají hodnot od – 2147483648 do + 268435456. Můžeme zde tedy najít i negativní hodnoty. Když si kategorie rozdělíme na negativní a pozitivní a opět najdeme minimální a maximální hodnoty získáme hodnoty na obrázku níže:

```
Min. negativní:  -2147483648
Max. negativní:  -18748192
Min. pozitivní:  1
Max. pozitivní:  268435456
```

Obrázek 6.11: *rozdělení hodnot v kategorických datech*

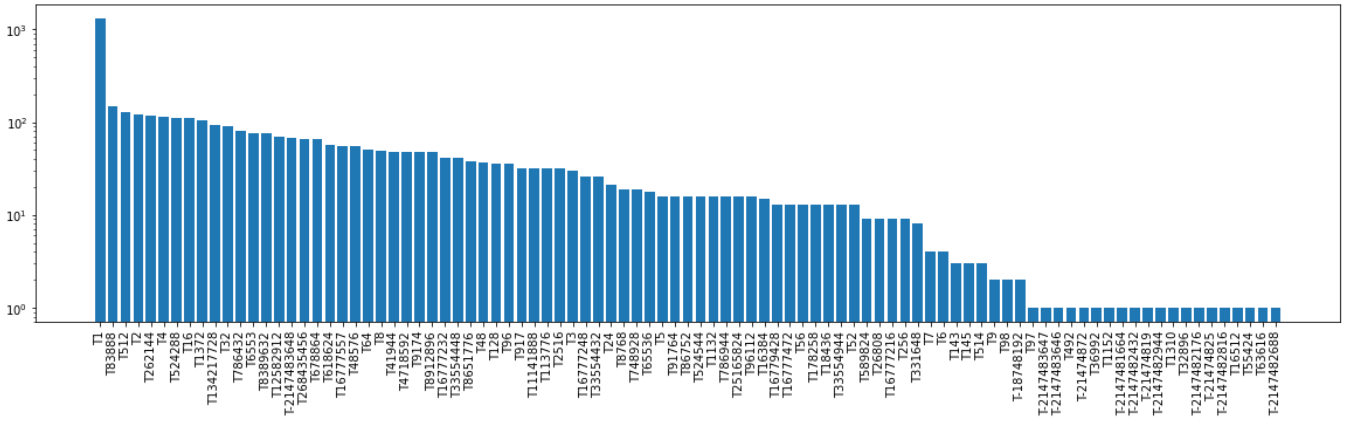
Tyto nové informace nám naznačují, že zpracování kategorického datasetu standardními metodami nebude možné. Počet kategorií je 93, což neumožňuje využít metodu *one-hot-encoding*, kdy bychom museli každému sloupci přiřadit nových 93 numerických vlastností pro rozpoznání kategorií. Tato metoda by vytvořila nový dataset, kde by bylo 199 113 sloupců. Velikost kategorických dat by neklesla, nýbrž výrazně narostla, což není ideální, pokud se snažíme data redukovat. Taktéž by zde vznikl nový problém nárůstu komplexnosti dat. Prediktivní modely by nedokázaly pracovat s datasetem takovýchto dimenzí. Navíc zde musíme počítat se skutečností, že jsou tyto analýzy prováděny pouze na trénovacích datech. Máme zde také testovací soubory, které mohou obsahovat mnohem více jednotlivých kategorií, se kterými bychom nemohli pracovat.

Další způsob transformace kategorických dat je využití číselné složky kategorií pro nové hodnoty. Narážíme ale na problém, že číselné hodnoty kategorií mají velký rozptyl. Prediktivní modely by tedy měly minimální informativní rozdíl mezi hodnotami T1 a T2. Jelikož jsou kategorie anonymní, není jasný význam jednotlivých kategorií a existuje velká pravděpodobnost, že jejich význam je velmi rozdílný. Po převedení kategorických hodnot na numerické by prediktivní model nerozpoznal téměř žádný rozdíl mezi některými kategoriemi, což není žádané, protože bychom riskovali velkou ztrátu informací.

6.4.2 **Vizualizace**

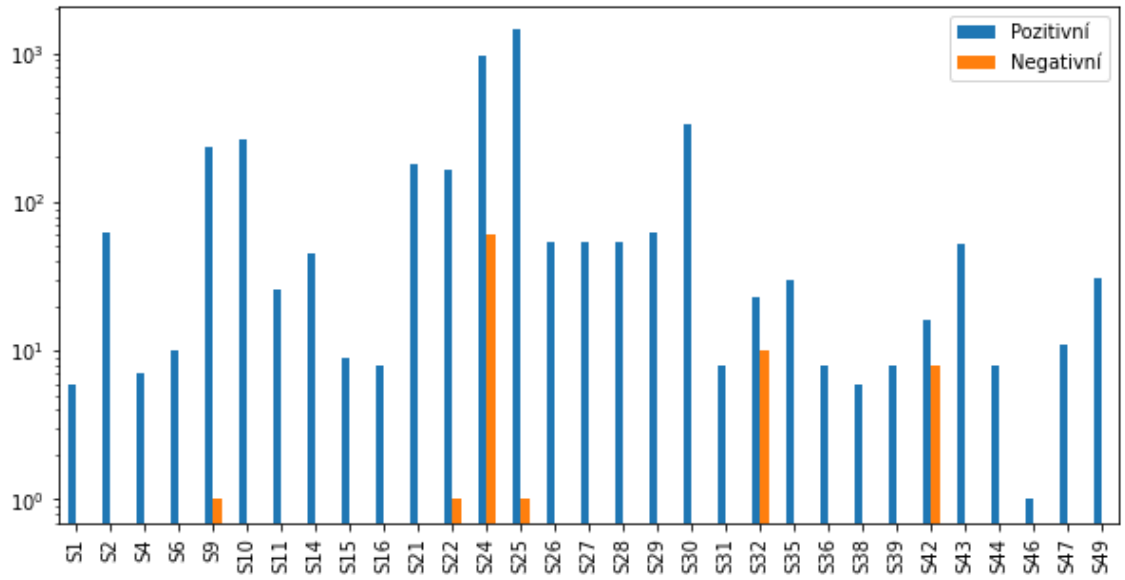
Jelikož jsme nedokázali najít metody pro redukci a transformaci kategorických dat, musíme data vizualizovat a pokusit se nalézt charakteristiky a vztahy v datech, které můžeme pak při výběru vlastností a tvorbě nových vlastností využít.

První graf (obr. 6.12) vyobrazuje výskyt jednotlivých kategorií napříč kategorickým datasetem. Pro lepší přehlednost je Y osa v logaritmickém měřítku. Z grafu je jasné, že nejčastější kategorií je kategorie je T1. Tato kategorie se v datasetu vyskytne přesně 922 krát, druhá nejčastější kategorie je T2 s výskytem 106 krát. Dále si můžeme všimnout, že kategorie s negativní číselnou hodnotou jsou velmi ojedinělé v kategorickém datasetu a celkově jich existuje pouze 13.



Obrázek 6.12: *výskyt kategorií v datasetu*

Další graf (obr. 6.13) zobrazuje výskyt pozitivních a negativních kategorií v každé stanici výrobní linky. Z grafu je jasné, že se negativní hodnoty začínají ukazovat až od stanice S9. Nejčastější negativní kategorické hodnoty vidíme ve stanicích S24, S32 a S42.



Obrázek 6.13: *výskyt pozitivních a negativních kategorií*

6.4.3 **Tvorba nových vlastností**

Základní metody pro úpravu a zpracování kategorických dat nemohou být na tento konkrétní kategorický dataset uplatněny. Logická analýza však odhalila jisté charakteristiky a vlastnosti těchto dat, které jsou ideální pro vytvoření nových vlastností. Tyto nové vlastnosti by měly zachovat informační hodnotu originálních dat, a přitom být jednoduše zpracovatelné pro prediktivní modely strojového učení.

Jedna z možností, jak můžeme využít kategorická data, je zaznamenat počet negativních kategorií za výrobní cyklus konkrétního produktu. Z grafu 6.13 je ale jasné, že se záporné kategorické hodnoty vyskytují jenom na určitých stanicích a jsou více ojedinělé, než pozitivní hodnoty. Není tedy potřebné, abychom počítali výskyt negativních hodnot pro každou stanici a každou vlastnost. Tímto se značně redukuje výpočetní čas a také se usnadní práce pro prediktivní modely.

Zároveň se tímto postupem ztratí minimální informační hodnota z originálního kategorického datasetu. Abychom docílili ještě větší redukce ztráty, zaznamenali jsme negativní hodnoty pro každou stanici zvlášť. Tím obcházíme nutnost sjednotit hodnoty do jedné finální sumy. Jako u redukovaných numerických dat, i zde ukládáme finální redukovaný kategorický dataset do binárního souboru *feather*:

	Id	S9_negatives	S22_negatives	S24_negatives	S25_negatives	S32_negatives	S42_negatives
0	4	0	0	0	0	0	0
1	6	0	0	0	0	0	0
2	7	0	0	0	0	0	0
3	9	0	0	0	0	0	0
4	11	0	0	0	0	0	0
...	...	...	...	...	...	...	...
1183742	2367490	0	0	0	0	0	0
1183743	2367491	0	0	0	0	0	0
1183744	2367492	0	0	0	0	0	0
1183745	2367493	0	0	0	0	0	0
1183746	2367495	0	0	0	0	0	0

Obrázek 6.14: *transformovaný kategorický dataset*

6.5 Časová data

Třetí a poslední kategorií dat v prozkoumávaném datasetu jsou časová data. Zde při analýze dat opět narážíme na problém s nejasností, jakou veličinu data kvůli své anonymitě popisují. Konvence pro pojmenovávání časových dat je mírně odlišná od té u numerických a kategorických. Každý sloupec časových dat odkazuje na sloupec v numerickém nebo časovém datasetu, kdy byla daná data naměřena. Například sloupec L0_S0_D1 nese informaci o tom, kdy byla hodnota ve sloupci L0_S0_F0 zaznamenána (Bosch 2016):

- Jako u předchozích kategorií i zde nalezneme *NaN* hodnoty, s kterými je nutné počítat.
- Hodnoty jsou numerického typu a nejsou normalizované. Můžeme zde nalézt hodnoty pohybující se řádově v desítkách, ale i stovkách a tisících.
- Počet vlastností v datasetu je 1157 včetně *Id* sloupce.

	Id	L0_S0_D1	L0_S0_D3	L0_S0_D5	L3_S51_D4257	L3_S51_D4259	L3_S51_D4261	L3_S51_D4263
0	4	82.24	82.24	82.24	NaN	NaN	NaN	NaN
1	6	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	7	1618.70	1618.70	1618.70	NaN	NaN	NaN	NaN
3	9	1149.20	1149.20	1149.20	NaN	NaN	NaN	NaN
4	11	602.64	602.64	602.64	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...
95	192	673.36	673.36	673.36	NaN	NaN	NaN	NaN
96	197	NaN	NaN	NaN	NaN	NaN	NaN	NaN
97	200	576.73	576.73	576.73	NaN	NaN	NaN	NaN
98	201	1214.60	1214.60	1214.60	NaN	NaN	NaN	NaN
99	205	1658.58	1658.58	1658.58	NaN	NaN	NaN	NaN

Tabulka 6.3: *tabulka vlastností časových dat (redukovaná)*

V časovém datasetu se nachází opět pouze 4 výrobní linky, 52 stanic a již zmíněných 1157 vlastností včetně *Id* sloupce. Toto zjištění spolu s informacemi získanými při prozkoumávání numerických a kategorických dat nám potvrzuje správnost prozkoumávání těchto charakteristik.

Jelikož se zde nacházejí *NaN* hodnoty, musí se zvolit vhodný způsob, jak se bude s těmito hodnotami zacházet. Před zvolením numerické reprezentace *NaN* hodnoty bylo zapotřebí zjistit maximální a minimální hodnoty, které se v časové kategorii dat nacházejí. Jako u ostatních kategorií, i zde narážíme na problém velikosti souboru a není možné využít standardní metody pro zpracování. Je tedy nutné využít metodu kouskování pro jakoukoliv práci s tímto datasetem nebo pro jeho prozkoumávání. Jednoduchý skript nám zjistí, že maximální hodnota v datasetu je 1691,67 a minimální hodnota je 0,71. Jelikož oba extrémy v datasetu jsou pozitivní, můžeme nahradit veškeré instance *NaN* hodnotou -1. Tato informace bude důležitá při tvorbě nových vlastností v časovém datasetu.

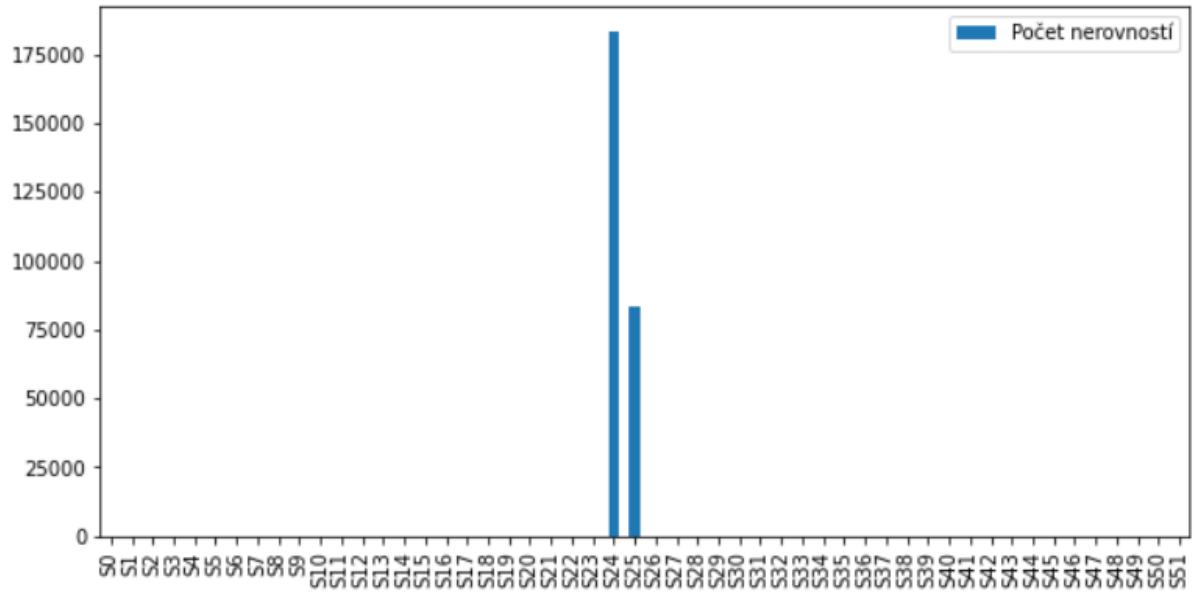
Při hledání extrémů v časovém datasetu pro jednotlivé stanice zjistíme podstatnou informaci. Minima a maxima jsou pro většinu vlastností v jednotlivých stanicích naprosto identická viz. obrázek níže:

```
Stanice S0:
Max. hodnoty vlastností: 82.24, nan, 1618.7, 1149.2, 602.64, 1331.66, nan, nan, 517.64, nan, ...
Min. hodnoty vlastností: 82.24, nan, 1618.7, 1149.2, 602.64, 1331.66, nan, nan, 517.64, nan, ...

Stanice S1:
Max. hodnoty vlastností: 82.24, nan, 1618.7, 1149.2, 602.64, 1331.66, nan, nan, 517.64, nan, ...
Min. hodnoty vlastností: 82.24, nan, 1618.7, 1149.2, 602.64, 1331.66, nan, nan, 517.64, nan, ...

Stanice S2:
Max. hodnoty vlastností: 82.24, nan, 1618.7, 1149.21, nan, nan, nan, nan, 517.64, nan, ...
Min. hodnoty vlastností: 82.24, nan, 1618.7, 1149.21, nan, nan, nan, nan, 517.64, nan, ...
```

Obrázek 6.15: *maximální a minimální hodnoty pro jednotlivé vlastnosti (redukované)*

Tento charakter dat nám prozrazuje, že veškerá data generovaná v jednotlivých stanicích byla přečtena ve stejný časový okamžik. To znamená, že například vlastnost L0_S0_F1 byla přečtena a uložena ve stejnou dobu jako všechny ostatní vlastnosti na stanici S0. Tato informace je velmi důležitá pro tvorbu nových vlastností a později je k tomu také využita.

Pro úplnou představu, zda se v datech nenacházejí stanice, které vystupují z tohoto trendu, si můžeme data vizualizovat. Graf níže (obr. č.21) je výstup programu, který porovnává minima a maxima v jednotlivých stanicích a hledá nesrovnalosti. Pokud program narazí na stanici, kde byla data z čidel generována v jiný časový okamžik, inkrementuje hodnotu pro danou stanici. Tento postup nám umožní vidět nesrovnalosti v jednotlivých stanicích časového datasetu.



Obrázek 6.16: *nerovnosti v jednotlivých stanicích*

Z grafu (obr. 6.16) je dále jasně rozpoznatelné, že stanice S24 a S25 jsou jediné stanice, kde byly nalezeny nesrovnalosti v maximálních a minimálních hodnotách. Jelikož je počet vzorků, u kterých byly tyto nesrovnalosti nalezeny, signifikantní, nelze je ignorovat. Je zapotřebí s touto informací počítat při tvoření nových vlastností časových dat.

6.5.1 **Tvorba nových vlastností**

V časových datech můžeme logickou dedukcí vytvořit nejvíce nových vlastností. Jelikož jsme opět omezeni velikosti paměti, nemůžeme pracovat s veškerými vlastnostmi a jsme nuceni redukovat dataset do zpracovatelné velikosti.

První nová vlastnost, kterou lze vytvořit pro časová data, je nalezení minimální hodnoty pro daný produkt. Jelikož se v našem případě jedná o časová data, tato hodnota reprezentuje čas, kdy byl daný výrobek zaznamenán na výrobní lince. Tato nová vlastnost je v datasetu nazvána jako *Start_time*, tedy čas, kdy začal výrobní cyklus daného produktu. Důvod pro vytvoření této nové vlastnosti je dedukce, že prediktivní modely úspěšně naleznou vzorce a relace mezi různými zahajovacími časy výroby. Další nová vlastnost, která je velmi jednoduchá k vytvoření, je čas ukončení výroby jednotlivých produktů. Tato vlastnost je pojmenována *Stop_time*. Abychom ještě posílili tento pár vlastností, je zcela logické vytvořit hodnotu reprezentující trvání výroby daného produktu. Hodnotu lze získat pouhým rozdílem vlastností *Start_time* a *Stop_time*. Tato nová vlastnost nese název *Production_time*.

Nyní můžeme zužitkovat získané informace z předchozí kapitoly, kde jsme zjistili anomálie u stanic S24 a S25. Abychom co nejvíce zamezili ztrátě informační hodnoty, je relevantní zpracovat nové vlastnosti *Start_time*, *Stop_time* a *Production_time* i jednotlivě pro tyto dvě stanice.

Poslední nové vlastnosti, které byly vytvořeny pro časová data je pár hodnot nesoucí informaci o první a poslední stanici, na které byl výrobek zaznamenán. Tento pár vlastností nese název *Start_station* a *Stop_station*. Na obrázku níže jsou vyobrazeny všechny nové vlastnosti časových dat:

S24_Start_time	S24_Stop_time	S25_Start_time	S25_Stop_time	Start_time	Stop_time	S24_Production_time	S25_Production_time	Production_time	Start_station	Stop_station
792.72	792.77	NaN	NaN	792.72	800.70	0.05	NaN	7.98	24	34
1025.63	1025.64	NaN	NaN	1025.63	1060.07	0.01	NaN	34.44	24	36
671.92	671.95	NaN	NaN	671.92	711.08	0.03	NaN	39.16	24	34
NaN	NaN	NaN	NaN	255.45	256.28	NaN	NaN	0.83	0	36
743.38	743.40	NaN	NaN	743.38	770.28	0.02	NaN	26.90	24	33
...	...	...	...	...	...	...	...	...	...	...
NaN	NaN	NaN	NaN	653.85	655.45	NaN	NaN	1.60	0	35
NaN	NaN	NaN	NaN	907.34	911.29	NaN	NaN	3.95	0	34
NaN	NaN	NaN	NaN	185.92	186.32	NaN	NaN	0.40	0	35
NaN	NaN	NaN	NaN	570.85	572.67	NaN	NaN	1.82	0	35
NaN	NaN	NaN	NaN	1412.80	1413.18	NaN	NaN	0.38	0	33

Obrázek 6.17: *nové vlastnosti časových dat*

6.6 **Trénovaní modelů**

Pro tvorbu prediktivních modelů byly využity modely pro klasifikační predikci z knihovny *sklearn*. V tomto kroku využijeme zredukovaná data z předchozích kapitol uložených do binárních souborů *feather*.

Hlavní modely zvolené pro prediktivní učení jsou *ExtraTreesClassifier*, BaggingClassifier a *RandomForestClassifier*. Tyto modely jsou jedny z komplexnějších prediktivních modelů, které využívají libovolně zvolený počet jednodušších prediktivních modelů. Počet jednoduchých modelů lze nastavit pomocí parametru n*_estimators*. Dále také všechny tři modely je možné trénovat s využitím paralelismu prostřednictvím knihovny *Dask*. Díky paralelismu bude možné modely trénovat několikanásobně rychleji než bez použití této metody.

K analýze a predikci dat byly otestovány také lineární modely. Jejich MCC skóre se však pohybovalo v záporných hodnotách, a proto nebyly dále nijak laděny nebo testovány.

6.7 Porovnání výsledků

Stránky Kaggle, jak již bylo zmíněno, uvádí u jednotlivých soutěžních datasetů rozsáhlé diskuse datových vědců o postupech a informacích vybraných dat. V této kapitole porovnáme vlastní výsledky se závěry datových vědců na stránkách Kaggle.

Stejně jako několik datových vědců i tato práce obsahuje vizualizace schémat výrobní linky. Zpracovatelé datasetu zveřejnili svůj zdrojový kód a vizualizace jsou volně dostupné. Jeden z nejpopulárnějších článků je *Shopfloor Visualization* (Chris 2016). Postup vizualizace tohoto autora je místy odlišný od vizualizace této práce. Vidíme zde klady i zápory, které s sebou nese autorův rozdílný postup. První rozdíl je zjevný: vizualizace autora Chris (2016) byla provedena pouze na 10 000 vzorcích, zatímco v této bakalářské práci byl zdrojový kód zefektivněn, aby v relativně nízkém čase mohl být použit na celý dataset. Dalším rozdílem ve vizualizaci dat je umístění jednotlivých uzlů. Chris (2016) využívá statické umístění uzlů oproti zdrojovému kódu této práce, který přiřazuje pozici jednotlivým uzlům dynamicky. Dynamické zpracování dat je více flexibilní a lze jej využít na jakýkoliv jiný dataset, který následuje konvenci pojmenovávání jednotlivých vlastností jako zkoumaný dataset firmy Bosch. Stejně jako u autora Chris (2016) i vizualizace v této bakalářské práci správně vizualizovala stanice S46 a S42 jako nepoužité. Grafické výstupy této bakalářské práce týkajících se schématu produkční linky také dále koresponduje s článkem *Manufacturing Floor Visualization* od datového vědce Maclean (2016).

Závěr, ke kterému jsme došli v kapitole 6.3.2 Vizualizace praktické části této bakalářské práce týkající se stanice S32 je podložen článkem *What's wrong with station 32* autora Delaney (2016). V tomto článku se můžeme dočíst, že stanice S32 má nejvyšší chybovost z celé produkční linky. Numerické výsledky této práce jsou identické s výstupy článku *What's wrong with station 32* autora Delaney (2016), grafické výstupy jsou velmi podobné.

Výstupy z kapitol ohledně selekce vlastností z numerických dat této bakalářské práce odpovídají těm v diskusích na stránkách Kaggle. V kapitole *6.3.3 Výběr vlastností* z obrázku 6.8 lze vidět, že nejčastěji zaznamenané vlastnosti selektivními algoritmy jsou ze stanice S24. Dále zde také můžeme vidět, že se zde objevuje jediná vlastnost stanice S32 jako jedna z nejvíce informativně hodnotných vlastností. Tyto informace odpovídá s výstupy článku *Expeditive exploration+models* on data od autora Laure (2016).

Závěry ohledně kategorických dat uvedené v kapitole 6.*4 Kategorická* data na základě logické analýzy odpovídají závěrům vítězného týmu soutěže, jejímž členy byly datoví vědci Gábor Fodor a Ash Hafez. Vítězný tým ve svém článku *1st place solution* (Fodor, Hafez 2016) uvádějí, že z kategorických dat bylo velmi složité extrahovat jakoukoliv informativně hodnotnou vlastnost. Průzkum kategorických hodnot uvedený v praktické části této bakalářské práce je dále také podložen výstupy z článku *Looking at Categorical Data* (Ronin 2016a) a také článkem *Categories and the power of 2* (Ronin 2016b).

Vytvořené nové vlastnosti pro kategorii časových dat v kapitole *6.5.1 Tvorba nových vlastností* odpovídá novým vlastnostem, které také využili první tým a jsou zaznamenané v článku *1st place solution* (Fodor, Hafez 2016). Informace obdržená při prozkoumávání časových dat o duplikátech v hodnotách u jednotlivých stanic je také podložena diskusemi *Same timestamp at many date columns, meaning?* (Nedelkoski 2016) a *Date Exploration (6min == 0.01)* (Fodor 2016).

Nejlépe hodnocený prediktivní model, který byl vytvořený v této bakalářské práci je *RandomForestClassifier* z privátním MCC skóre 0,24704 a veřejným MCC skóre 0,23049 (důkaz těchto hodnot je na obrázku 6.18). Veřejné MCC skóre znamená vyhodnocení MCC pouze na 30 % datasetu a privátní MCC skóre je vyhodnocováno na zbylých 70 %. Vyhodnocení soutěže se odvíjí od privátního MCC skóre. Výherní tým dosáhnul privátního MCC 0.52401 z celkem 1370 týmů. Skóre prediktivního modelu, který byl sestrojen v praktické části této bakalářské práce by se umístilo na 669 příčku.

Obrázek 6.18: *privátní a veřejné MCC skóre*.

Závěr

Velká data jsou neustále vyvíjejícím se tématem v datové vědě a pro správné chápání velkých dat je nezbytné mít přehled o novodobých trendech z oblasti nástrojů pro analýzu tohoto druhu dat. Široké spektrum nejrůznějších nástrojů popsaných v této práci podtrhuje složitost a komplexitu problematiky velkých dat. Cílem této bakalářské práce je seznámení čtenáře s nástroji a postupy pro správnou analýzu velkých dat a následná aplikace a prokázání funkčnosti těchto nástrojů na konkrétním datasetu. Čtenář by měl mít po přečtení této práce k dispozici přehled nejrůznějších nástrojů a vědomostí o tom, proč a kde dané nástroje použít. Tímto se podařilo naplnit definované cíle práce.

Teoretická část podrobněji popisuje jednotlivé nástroje, které jsou potřebné v každé fázi cyklu zpracování velkých dat. Nalezneme zde kapitoly popisující novodobé trendy v nástrojích pro sběr velkých dat. Jedná se o protokoly a softwarové nástroje, které se běžně využívají pro shromažďování velkých dat. Dále jsou v této práci rozpracovány druhy uložišť a základní koncepty této problematiky, jako datové modely nebo formáty dat. Čtenář se také seznámí s nástroji pro analýzu velkých dat a jejich výhody. Následně jsou v této práci popsány algoritmy a metody zpracování velkých dat strojovým učením, včetně vysvětlení jejich základních charakteristik.

Pro účely projektu byl zvolen dataset, na kterém bylo demonstrováno správné využití vybraných nástrojů. Zvolený dataset vygenerovaný produkční linkou firmy Bosch byl součástí soutěže na stránkách Kaggle. Jedná se o velmi komplexní dataset, který slouží jako vhodná ukázka k otestování zmíněných nástrojů a metod. Cílem analýzy datasetu byla predikce kvality výroby jednotlivých produktů na základě hodnot z jejich výrobního cyklu.

Pro zpracování vybraného datasetu byl zvolen programovací jazyk *Python* a bylo využito portfolio knihoven, které jsou standardem pro datovou vědu v programovacím jazyce *Python*. Knihovny *matplotlib* a *networkx*, byly využity pro generování grafických výstupů a pro načítání a manipulaci s daty byla zvolena knihovna *Pandas*. Z knihovny *sklearn* byly použity algoritmy pro selekci vlastností a také pro sestrojení a následné trénování prediktivních modelů strojového učení. Pro distribuci jednotlivých procesů mezi všechna jádra procesoru koncového zařízení byly využity knihovny *Dask* a *joblib*.

Vybraný dataset se skládá ze tří hlavních datových kategorií: numerických, kategorických a časových. Každá kategorie je pečlivě prozkoumána a výstupy této analýzu jsou zaznamenány v oddělených kapitolách. U jednotlivých kategorií jsou popsány metody pro transformaci, modifikaci a práci s těmito daty.

Z analýzy dat se zjistilo, že stanice S32 je nejvíce kritickou ve výrobním cyklu produktu a má na něj největší vliv. Došli jsme k závěru, že kategorická data mají nejmenší informativní hodnotu o výrobě daných produktů. Zatímco časová a numerická data obsahují informačně silné vlastnosti.

Z datasetu se podařilo extrahovat informace o struktuře výrobních linek, což nám nastínilo další důležité informace. Tyto informace se využily k tvorbě nových vlastností, jako je například doba výroby nového produktu.

Jednotlivé závěry praktické části této práce jsou porovnávány se závěry datových vědců, kteří v rámci soutěže na platformě Kaggle taktéž zpracovávali stejný dataset. Výsledky této práce korespondují s jejich výstupy. Závěry této práce také popisují přidanou hodnotu určitých algoritmů vytvořených specificky pro tuto práci. Finální prediktivní model na transformovaném datasetu dokázal predikovat kvalitu výrobku s přesností 0.24704 dle MCC.

Další vývoj tohoto projektu by se měl zaměřit na nalezení silnějšího výpočetního zařízení, na kterém bude prováděna analýza konkrétního datasetu. Tato změna by umožnila otestování a modelování složitějších algoritmů strojového učení a také využití dalších metod pro transformaci a manipulaci s daty. Použité postupy a nástroje této práce je možné otestovat na jiných datech sbíraných v moderních digitalizovaných výrobních procesech. Zdrojové kódy tohoto projektu lze využít na jakýkoliv jiný dataset, který následuje konvenci pojmenovávání jednotlivých vlastností jako zkoumaný dataset firmy Bosch.

Použitá literatura

[1] ADLER, Joseph. R in a Nutshell: A Desktop Quick Reference [online]. 1005 Gravenstein Highway North, Sebastopol, CA 95472.: O'Reilly Media, 2010 [cit. 2020-12-18]. ISBN 978-0-596-80170-0. Dostupné z: http://web.udl.es/Biomath/Bioestadistica/R/Manuals/r_in_a_nutshell.pdf

[2] APACHE SOFTWARE FOUNDATION. INTRODUCTION. Apache Kafka: A distributed streaming platform [online]. 2017 [cit. 2020-12-15]. Dostupné z: https://kafka.apache.org/intro

[3] APACHE SPARK. MLlib: is Apache Spark's scalable machine learning library. [online]. 2018 [cit. 2020-12-18]. Dostupné z: https://spark.apache.org/mllib/

[4] APACHE SOFTWARE FOUNDATION. Flume Developer Guide. Flume Apache [online]. 2019 [cit. 2020-12-17]. Dostupné z: https://flume.apache.org/releases/content/1.7.0/FlumeDeveloperGuide.html

[5] ALTO, Valentina. Unsupervised Learning: K-means vs Hierarchical Clustering. Towards Data Science [online]. 2019 [cit. 2021-03-13]. Dostupné z: https://towardsdatascience.com/unsupervised-learning-k-means-vs-hierarchical-clustering-5fe2da7c9554

[6] BANK OF AMERICA. AMQP [online]. [cit. 2020-12-12]. Dostupné z: https://www.amqp.org/sites/amqp.org/files/amqp.pdf

[7] BOSCH. Bosch Production Line Performance: Reduce manufacturing failures. Kaggle [online]. Bosch, 2016 [cit. 2021-03-25]. Dostupné z: https://www.kaggle.com/c/bosch-production-line-performance/overview

[8] BROWNLEE, Jason. An Introduction to Feature Selection. Machine Learning Mastery [online]. 2014 [cit. 2021-03-13]. Dostupné z: https://machinelearningmastery.com/an-introduction-to-feature-selection/

[9] CAVANILLAS, José María, Edward CURRY a Wolfgang WAHLSTER, ed. New Horizons for a Data-Driven Economy [online]. Cham: Springer International Publishing, 2016 [cit. 2020-12-07]. ISBN 978-3-319-21568-6. Dostupné z: doi:10.1007/978-3-319-21569-3

[10] CHEN, Min, Shiwen MAO a Yunhao LIU. Big Data: A Survey [online]. New York: Springer Science+Business Media, 2014 [cit. 2020-12-18]. Dostupné z: doi:10.1007/s11036-013-0489-0

[11] CHRIS, "Gingerman." Shopfloor Visualization. Kaggle [online]. 2016 [cit. 2021-4-27]. Dostupné z: https://www.kaggle.com/gingerman/shopfloor-visualization

[12]DELANEY, Jeff. What's Wrong with Station 32. Kaggle [online]. 2016 [cit. 2021-4-27]. Dostupné z: https://www.kaggle.com/jeffd23/what-s-wrong-with-station-32

[13]DE MAURO, Andrea, Marco GRECO a Michele GRIMALDI. A formal definition of Big Data based on its essential features. Library Review [online]. 2016, 65(3), 122-135 [cit. 2020-11-28]. ISSN 0024-2535. Dostupné z: doi:10.1108/LR-06-2015-0061

[14]DEAN, Jared. Big data, data mining and machine learning: value creation for business leaders and practitioners. Hoboken, New Jersey: John Wiley and Sons, [2014]. ISBN 978-1-118-92069-5.

[15]EMC EDUCATION SERVICES. Data Science and Big Data Analytics: Discovering, Analyzing, Visualizing and Presenting Data [online]. 2015 [cit. 2020-12-19]. ISBN 978-1-118-87613-8. Dostupné z: https://www.wiley.com/en-us/Data+Science+and+Big+Data+Analytics%3A+Discovering%2C+Analyzing%2C+Visualizing+and+Presenting+Data-p-9781118876138

[16]ERL, Thomas, Wajid KHATTAK a Paul BUHLER. Big Data Fundamentals: Concepts, Drivers & Techniques. Arcitura Education, 2015. ISBN 0-13-429107-7.

[17]Elite Data Science. Dimensionality Reduction Algorithms: Strengths and Weaknesses [online]. 2019 [cit. 2021-03-13]. Dostupné z: https://elitedatascience.com/dimensionality-reduction-algorithms

[18]FODOR, Gábor a Ash HAFEZ. 1st place solution. Kaggle [online]. 2016 [cit. 2021-4-27]. Dostupné z: https://www.kaggle.com/c/bosch-production-line-performance/discussion/25434

[19]FODOR, Gábor. Date Exploration (6min == 0.01). Kaggle [online]. 2016 [cit. 2021-4-27]. Dostupné z: https://www.kaggle.com/gaborfodor/notebookd19d11e4f2

[20]FRANK, Eibe et al. Weka-A Machine Learning Workbench for Data Mining. Maimon O., Rokach L. (eds) Data Mining and Knowledge Discovery Handbook [online]. Boston: Springer, 2009, Dostupné z: doi:https://doi.org/10.1007/978-0-387-09823-4_66.

[21]HAPNER, Mark, Rich BURRIDGE a Rahul SHARMA. ORACLE. JMS Specification [online]. [cit. 2020-12-14]. Dostupné z: https://docs.oracle.com/cd/E19957-01/816-5904-10/816-5904-10.pdf

[22]HOFFMAN, Steve. Apache Flume: Distributed Log Collection for Hadoop. 2nd edition. Livery Place 35 Livery Street Birmingham B3 2PB, UK: Packt Publishing, 2015. ISBN 978-1-78439-217-8.

[23]HUNTER, J. D. Matplotlib: A 2D Graphics Environment [online]. Computing in Science & Engineering, vol. 9, no. 3, pp. 90-95, 2007. [cit. 2020-12-18]. Dostupné z: https://matplotlib.org/citing.html

[24]IMB 2013: IBM-bigdata-platform: IBM Big Data Strategy [online]. [cit. 2020-12-12]. Dostupné z: http://public.dhe.ibm.com/software/data/sw-library/big-data/ibm-bigdata-platform-19-04-2012.pdf

[25]INTELLIPAAT. What is Apache Storm? Intellipaat [online]. 2016 [cit. 2020-12-15]. Dostupné z: https://intellipaat.com/blog/what-is-apache-storm/

[26]Java and Big Data: why Big Data projects can't do without Java. CodeGym [online]. 2020 [cit. 2020-12-18]. Dostupné z: https://codegym.cc/groups/posts/278-java-and-big-data-why-big-data-projects-cant-do-without-java

[27]KUBAT, Miroslav. An Introduction to Machine Learning [online]. Cham: Springer International Publishing, 2017 [cit. 2021-03-10]. ISBN 978-3-319-63912-3. Dostupné z: doi:10.1007/978-3-319-63913-0

[28]LAURAE. Expeditive exploration+models on data. Kaggle [online]. 2016 [cit. 2021-4-27]. Dostupné z: https://www.kaggle.com/c/bosch-production-line-performance/discussion/22909

[29]MACLEAN, Cole. Manufacturing Floor Visualization. Kaggle [online]. 2016 [cit. 2021-4-27]. Dostupné z: https://www.kaggle.com/c/bosch-production-line-performance/discussion/23319

[30]MÜLLER, Andreas C. a Sarah GUIDO. Introduction to Machine Learning with Python: A Guide for Data Scientists [online]. 1005 Gravenstein Highway North, Sebastopol, CA 95472.: O'Reilly Media, 2016 [cit. 2020-12-18]. ISBN 978-1-449-36941-5.

[31]MONSON-HAEFEL, Richard a David A. CHAPPELL. Java Message Service [online]. A. Chappell, 2001 [cit. 2020-12-14]. ISBN 0-596-00068-5.

[32]NEXUS INTEGRA. Why is Big Data the core of the 4.0 industry? [online]. 2020 [cit. 2021-03-20]. Dostupné z: https://nexusintegra.io/big-data-industry-4-0/

[33]NEDELKOSKI, Sasho. Same timestamp at many date columns, meaning ? Kaggle [online]. 2016 [cit. 2021-4-27]. Dostupné z: https://www.kaggle.com/c/bosch-production-line-performance/discussion/24826

[34]NUMPY. ECOSYSTEM. Numpy [online]. 2020 [cit. 2020-12-18]. Dostupné z: https://numpy.org/

[35]NICHOLSON, Chris. A Beginner's Guide to Neural Networks and Deep Learning [online]. Pathmind Inc.., 2020 [cit. 2021-03-10]. Dostupné z: https://wiki.pathmind.com/neural-network#gradient

[36]OASIS Advanced Message Queuing Protocol (AMQP) Version 1.0. Oasis [online]. 2012 [cit. 2020-12-12]. Dostupné z: http://docs.oasis-open.org/amqp/core/v1.0/os/amqp-core-complete-v1.0-os.pdf

[37]ORACLE 2012: An Enterprise Architect's Guide to Big Data [online]. [cit. 2020-12-12]. Dostupné z: https://www.oracle.com/technetwork/topics/entarch/articles/oea-big-data-guide-1522052.pdf

[38]PADHY, Rabi Prasad. Big Data Processing with Hadoop-MapReduce in Cloud Systems. International Journal of Cloud Computing and Services Science (IJ-CLOSER) [online]. 2013 [cit. 2020-12-17]. ISSN 2089-3337.

[39]PRAJAPATI, Vignesh. Big Data Analytics with R and Hadoop. Livery Place 35 Livery Street Birmingham B3 2PB, UK: Packt Publishing, 2013. ISBN 978-1-78216-328-2.

[40]PROJECT JUPYTER. The Jupyter Notebook. Project Jupyter [online]. Last updated Wed, Nov 18, 2020 [cit. 2020-12-18]. Dostupné z: https://jupyter.org/

[41]ROUSE, Margaret (2014a). Object storage. Techtarget [online]. 2014 [cit. 2020-12-07]. Dostupné z: https://searchstorage.techtarget.com/definition/object-storage

[42]ROUSE, Margaret (2014b). Block storage. Techtarget [online]. 2014 [cit. 2020-12-07]. Dostupné z: https://searchstorage.techtarget.com/definition/block-storage

[43]ROMAN, Victor. Unsupervised Learning: Dimensionality Reduction. Towards Data Science [online]. 2019 [cit. 2021-03-13]. Dostupné z: https://towardsdatascience.com/unsupervised-learning-dimensionality-reduction-ddb4d55e0757

[44]RONIN. Looking at Categorical Data. Kaggle [online]. 2016a [cit. 2021-4-27]. Dostupné z: https://www.kaggle.com/rdslater/looking-at-categorical-data

[45]RONIN. Categories and the power of 2. Kaggle [online]. 2016b [cit. 2021-4-27]. Dostupné z: https://www.kaggle.com/c/bosch-production-line-performance/discussion/24018

[46]STROHBACH, Martin, Jörg DAUBERT, Herman RAVKIN a Mario LISCHKA. Big Data Storage. CAVANILLAS, José María, Edward CURRY a Wolfgang WAHLSTER, ed. New Horizons for a Data-Driven Economy [online]. Cham: Springer International Publishing, 2016, 2016, s. 119-141 [cit. 2020-12-04]. ISBN 978-3-319-21568-6. Dostupné z: doi:10.1007/978-3-319-21569-3_7

[47]SAYAD, Saed. Artificial Neural Network [online]. 2021 [cit. 2021-03-10]. Dostupné z: https://www.saedsayad.com/artificial_neural_network.htm

[48]SHAMSUDDIN, Siti Mariyam, Siti YUHANIZ a Nur Farhana HORDRI. Deep Learning and Its Applications: A Review [online]. Univerzita Teknologi Malaysia, Kuala Lumpur, 2016 [cit. 2021-03-10].

[49]TIWARI, Ayush. Apache Storm: Architecture [online]. 2017 [cit. 2020-12-15]. Dostupné z: https://dzone.com/articles/apache-storm-architecture

[50]TUTORIALS POINT. Hadoop – MapReduce. Tutorials Point [online]. 2020 [cit. 2020-12-17]. Dostupné z: https://www.tutorialspoint.com/hadoop/hadoop_mapreduce.htm

[51]TÝM VÝVOJÁŘŮ PANDAS, J. D. Pandas [online]. Zenodo, 2020 [cit. 2020-12-18]. Dostupné z: https://doi.org/10.5281/zenodo.3509134

[52]VENKATESH, Prasanna a Nirmala S. The new way to handle big data. Open Source Foru [online]. 2012 [cit. 2020-12-10]. Dostupné z: . http:// www.opensourceforu.com/2012/01/newsql-handle-big-data/

[53]VINKA, Elin a Lovisa JOHANSSON. Apache Kafka: Beginners Guide [online]. Book version: 1.1. 2019 [cit. 2020-12-15]. Dostupné z: https://www.cloudkarafka.com/files/PtgDYAyuBn3fcJBZV/apache-kafka-beginner-guide.pdf

[54]WANG, Lidong a Cheryl Ann ALEXANDER. Machine Learning in Big Data. International Journal of Mathematical, Engineering and Management Sciences [online]. Vol. 1, No. 2, 52–61, 2016 [cit. 2020-12-18]. ISSN 2455-7749. Dostupné z: doi:10.33889/IJMEMS.2016.1.2-006

[55]WITTEN, Ian et al. The WEKA data mining software: An update. ACM SIGKDD Explorations Newsletter [online]. 2009 [cit. 2020-12-17]. Dostupné z: doi:10.1145/1656274.1656278

[56]WITKOWSKI, Krzysztof. Internet of Things, Big Data, Industry 4.0 – Innovative Solutions in Logistics and Supply Chains Management. Procedia Engineering [online]. 2017, 182, 763-769 [cit. 2021-03-20]. ISSN 18777058. Dostupné z: doi:10.1016/j.proeng.2017.03.197

[57]YU, Shui a Song GUO, ed. Big Data Concepts, Theories, and Applications [online]. Cham: Springer International Publishing, 2016 [cit. 2020-12-04]. ISBN 978-3-319-27761-5. Dostupné z: doi:10.1007/978-3-319-27763-9

[58]ZHELEV, Svetoslav a Anna ROZEVA. Data analytics and machine learning with Java [online]. In: . 2018, s. 060020- [cit. 2021-03-10]. Dostupné z: doi:10.1063/1.5082135

Seznam příloh

Příloha A: *Program pro prozkoumávání datasetu:* prozkoumavani.ipynb

Příloha B: *Program pro selekci numerických vlastností:* selekce.ipynb

Příloha C: *Program pro trénování modelů a tvorbu nových vlastností:* modely_tvorba.ipynb

Příloha D: *Kategorie pro unikátní kategorie v datasetu:* unikatni_kategorie.json

Příloha E: *Cesty produktů v datasetu:* cesty_produktu.txt

Příloha F: *Kategorie v datasetu:* kategorie.txt

Příloha G: *Vybrané vlastnosti numerického datasetu:* vybrane_vlastnosti.txt

Příloha H: *Nejčastěji vybrané numerické vlastnosti:* vybrane_vlastnosti_n2.txt

Příloha I: *Chybovosti jednotlivých stanic:* stanice_chybovosti.csv

Příloha J: *Chybovosti jednotlivých stanic generované druhým algoritmem:* stanice_chybovosti_rychle.csv

Příloha K: *Výsledky nejlepšího prediktivního modelu:* submission_final.csv